Hybrid DoS/DDoS Intrusion Detection System Using Machine Learning Frameworks

Sophia Zhu

Advisor: Suleyman Uludag

November 18, 2021

Abstract

The three goals of a secure network are confidentiality, integrity, and availability. DoS (Denial of Service)/DDoS (Distributed Denial of Service) attacks are attacks that aim to make data, hosts, networks, or other online resources unavailable to legitimate users through the manipulation of multiple sources. These attacks can be catastrophic to enterprise networks if these networks do not have cost-effective and timely detection solutions to detect and mitigate these attacks. In previous works, various machine learning methodologies have been proven instrumental in attack detection. In this paper, we compared and implemented machine learning algorithms, including multiple linear regression, decision tree, and support vector machine, to realize a DoS/DDoS hybrid intrusion detection system based on two well-known datasets: *KDD-CUP 1999* and *CICIDS-2017*. The performance of each algorithm is compared and analyzed. We propose a DoS/DDoS hybrid intrusion detection system which integrates various machine learning algorithms for best efficiency. In specifics, the system is composed of three key parts: feature reduction, network anomaly detection, and signature-based classification. As a result, a classification accuracy score of 99.98% is reached for both datasets.

Keywords— Cybersecurity, DoS, DDoS, Machine learning, Hybrid Intrusion Detection System

Contents

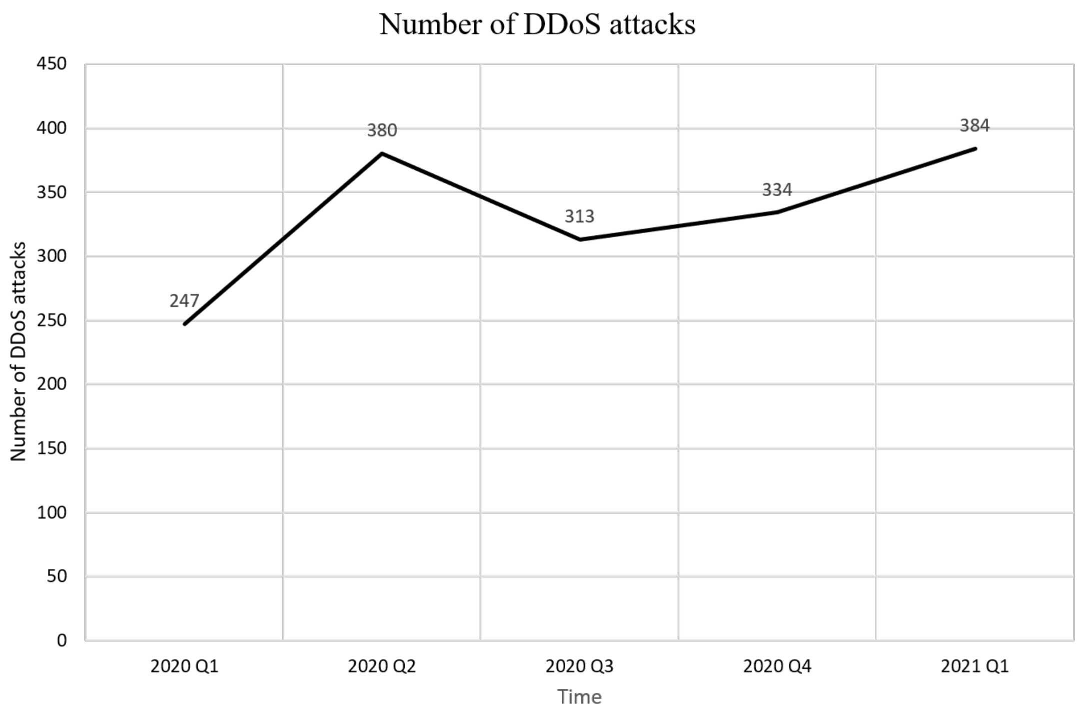


Figure 1: Increasing number of DDoS attack in recent years, with volumetric attacks being especially significant [1].

1 Introduction

In 2016, October 21st, a series of DDoS (Distributed Denial of Service) attacks were conducted against Dyn, an Internet infrastructure company that provides DNS (Domain Name Server) service. The first wave of attacks arrived at 7 AM, and Dyn resolved the attack in about two hours without causing too much of an issue. However, when the next wave of the DDoS attack arrived, the situation became out of control. Malicious pings were sent from millions of spoofed, distributed IP addresses. Dozens of websites became unavailable to users as the IP addresses of their web servers were no longer able to be matched with the users' inputs (URL) from web-browsers. The consequence was huge. Since Dyn provides services to mostly east-coast US, those living along the east-coast were affected the greatest; those living on the opposite side of America, as well as those living on the opposite side of the planet, all experienced difficulties while accessing websites and services online. This attack was based on the malware Mirai, and the attackers quickly controlled a private network with millions of host machines without legal acknowledgements, all sending requests to Dyn constantly and crushing its system. The Dyn attack was one of the worst cyberattack situations in history, in which the security of networks were severely violated by DDoS attacks.

With respect to Figure 1, DDoS attacks are becoming more and more common. As enterprises become increasingly aware of their network securities, the need for algorithms related to timely detection and mitigation with one of the most destructive attacks, the Distributed Denial-of-Service attack, has also risen drastically. Such rising demand is especially true in small enterprises; the high costs of private network protection services or the lack of resources cause many to be unprotected from network attacks.

This paper is structured as follows. Section 2 introduces the related work. Section 3 provides a brief review of the application of the supervised machine learning algorithms and the measuring metrics of accuracy. Section 4 discusses the basic implementation framework of the programs in this paper. Section 5 is a thorough explanation of the experimentation details and the overall evaluation of the programs. Section 6 describes the future works. Section 7 summarizes our major conclusions.

2 Related Work

There have been many efforts in the field of network intrusion detection system. This paper is inspired by analytical techniques and background information from several of those papers.

Sambangi [2] divides DoS attacks into three classes: protocol attacks, volumetric attacks, and application layer attacks.

1. In protocol attacks, the attacker floods the victim's machines with constant requests or pings, so that they can no longer respond to each request at such fast speeds. Protocol attacks are measured in packets per second.

2. In volumetric attacks, the attacker focuses on saturating the victim's bandwidth, and thus destroying connections to the victim. Volumetric attacks can consume a large amount of the victim's server resources by sending ICMP pings or UDP broadcast messages, which do not require a response. Volume-based attacks are measured in bits per second.

3. In application layer attacks, the attacker focuses on occupying and exhausting servers. The attacker takes advantage on the HTTP layer, identifies the slowest or most power-consuming part on a site or service, and uses HTTP GET/POST to flood the victim's servers to process the HTTP requests. Application layer attacks are measured in requests per second.

Each class of attack accounts for deviation from clean Internet connections in different attributes. Hence, we must use proprietary machine learning algorithms to classify these attacks.

Ahmed's [3] literature compares different types of machine learning algorithms' effectiveness with an increasing data size, considering both the training and detection times. Several classification-based training algorithms are used, and their detection time versus size of input data samples shows that the decision tree algorithm is most efficient in finding network anomalies for data sizes between 100 kB and 3400 kB. The decision tree algorithm also demonstrates excellent accuracy when tested on the *CAIDA DDoS-2007* [4] dataset, with a score of 99% [5] in Cvitić's paper [5], which also demonstrates the growing trend of DDoS attacks since 2000 and the emergent need for an efficient solution. In addition, authors in [6] and [7] used the decision tree algorithm to prove its effectiveness on detecting all network anomalies in *KDD-CUP 1999* [8]. Thus, in this research, we use the conclusions from the above related works.

Bouzida [9] makes attempts to merge the K-nearest neighbors and decision tree algorithms to generate a more accurate model for the *KDD-CUP 1999* dataset. The new model is quite successful despite a drawback on the classification of minority labels due to a lack of data samples. Singh [10] proposes a new SVM-based algorithms, named iSVM, also for classification on the *KDD-CUP 1999.*

On the other hand, Sambangi and Gondi's [2] paper uses a regression-based model rather than a classification-based model to detect DoS network anomalies in *CICIDS-2017* [11]. [12] stresses the importance of the conversion from categorical data labels to numerical, continuous values, for categorical data are generally unreadable for most regression-based training algorithms. Authors in [13] test several classification-based supervised machine learning algorithms on *CICIDS-2017*, with decision trees and naïve Bayes being two of the most outstanding algorithms. The result obtained from this paper corresponds mostly to the referenced papers using *KDD-CUP 1999.* Both datasets are widely used in literature. Allagi [14] uses support vector

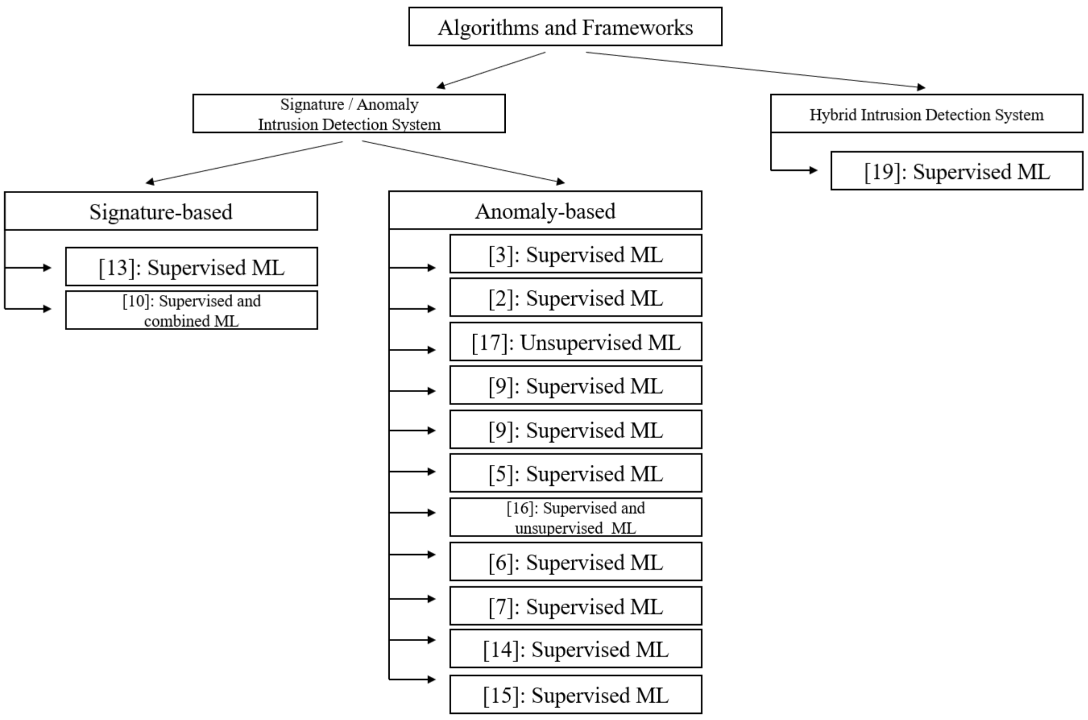


Figure 2: Related works displayed in a tree by NIDS and machine learning algorithm types.

machine for DoS attack detection in *CICIDS-2017*, also obtaining high accuracy scores.

Faisal [15] shows how loosely associated minority classes negatively impact the accuracy score of both support vector machine and random forest algorithms. As a result, the random forest algorithm performs better at identifying anomalies from the benign class, while support vector machine fails to acknowledge the huge variability between the abnormal class and the benign class, unable to produce satisfying accuracy while performing AIDS works.

Devi and Abualkibash [16] compare up to eight supervised machine learning algorithm's accuracy scores on *KDD-CUP 1999* and conclude that the best classification-based algorithm is AdaBoost, with low false alarm rate and high detection rate, while the K-nearest neighbor algorithm produces the best accuracy score. In this piece of work, the AdaBoost algorithm is built from weak classifiers of decision stumps, decision trees with one node and two leaves. Entropy-based random forests produce an even higher accuracy score and lower false alarm rate.

Wang and Bettiti's [17] paper provides insight on the transformation of features using principal component analysis, which aims to minimize the deviation from real values. The technique is shown to have significant effectiveness against network intrusion detection. Davis [18] provides detailed analysis on the features of *KDD-CUP 1999.*

[19] provides the most inspiration to this research. The author proposes a hybrid framework using decision tree SIDS and SVM AIDS to detect zero-day attacks and append their features back to the SIDS. The hybrid system is delicately designed based on the idea of appending each unknown attack type into the classification stage (SIDS). However, since this paper focuses on DoS/DDoS attack classification, we propose a differently structured hybrid system.

	Dataset	Feature Reduction	Algorithms	NIDS Type
Ahmed et al. [3]	*KDD-CUP 1999*; *UNSW-NB-15*	PCA Feature Extraction	Decision Tree; Naïve Bayes; Random Forest	AIDS
Sambangi and Gondi [2]	*CICIDS-2017*	IG Feature Selection	Multiple Linear Regression	AIDS
Wang and Battiti [17]	*KDD-CUP 1999*	PCA Feature Extraction	Clustering	AIDS
Bouzida et al. [9]	*KDD-CUP 1999*	PCA Feature Extraction	KNN; Decision Tree	AIDS
Cvitić et al. [5]	*CAIDA DDoS 2007*; *DARPA 2000*		Decision Tree; KNN; SVM; Naïve Bayes	AIDS
Devi and Abualkibash [16]	*KDD-CUP 1999*	Feature Extraction	Logistic Regression; Decision Tree; KNN; SVM; Random Forest; Multi-Layer Perception; AdaBoost; Naïve Bayes	AIDS
Kurniabudi et al. [13]	*CICIDS-2017*	Feature Reduction	Random Forest; Naïve Bayes; Decision Tree; Bayes Network	SIDS
Joong-Hee Lee et al. [6]	*KDD-CUP 1999*		Decision Tree	AIDS
Nancy Awadallah Awad [7]	*KDD-CUP 1999*		Decision Tree; Naïve Bayes; SVM; KNNs; Decision Table; ANN	AIDS
Ansam Khraisat et al. [19]	*NSL-KDD*		SVM; Decision Trees	HIDS (including zero-day attacks)
Singh et al. [10]	*KDD-CUP 1999*		SVM; iSVM	SIDS
Shridhar Allagi et al. [14]	*CICIDS-2017*		SVM	AIDS
Faisal Saleem Alraddadi et al. [15]	*UNSW-NB-15*; *KDD-CUP 1999*		SVM; Random Forest	AIDS
Zhu	*KDD-CUP 1999*; *CICIDS-2017*	PCA Feature Extraction	Decision Tree; Multiple Linear Regression; SVM	HIDS

Table 1: Table containing the information contained in each referenced paper, with the last row indicating the position this paper will take.

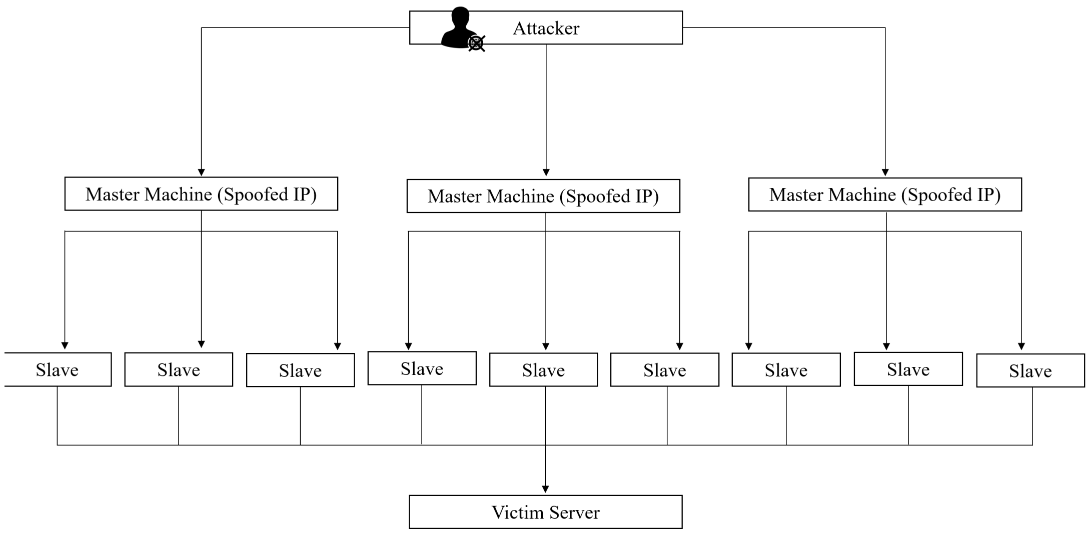

Attacker
Master Machine (Spoofed IP)
Master Machine (Spoofed IP)
Master Machine (Spoofed IP)
Slave
Slave
Slave
Slave
Slave
Slave
Slave
Slave
Slave
Victim Server

Figure 3: Structure of a DDoS attack

3 Background

Among the three fundamental characteristics of a secure network, confidentiality, integrity, and availability, DoS attacks target the availability of an online service. DDoS (Distributed Denial of Service) attack is formally defined as a malicious attempt to make an online service unavailable to legitimate users through compromised network systems. In Figure 3, the fundamental structure of a DDoS attack is shown. Attackers uses spoofed IP addresses to create an illusion that he is a legitimate user. In a DDoS web, the attacker controls a botnet of compromised machines to flood the victim's machine with packets or pings. By manipulating zombie master machines, the attacker is able to communicate and infect other machines in a network, turning them into slave machines. The floods of traffic can cause devastating consequences in a short period of time, so the time needed for accurate detection and classification of DoS/DDoS attacks should also be considered as an extremely important aspect in this research. The use of the machine learning algorithms involves the concepts introduced in this section.

3.1 IDS

Network Intrusion-detection systems (NIDS) are devices or applications that are used to detect or mitigate malicious activities in a network. NIDS can be classified into three main types: anomaly-based intrusion detection systems (AIDS), signature-based intrusion detection systems (SIDS), and hybrid intrusion detection systems (HIDS).

In an AIDS, data is either predicted to be benign or different from benign (abnormal). An AIDS is mostly used for monitoring network conditions and detecting anomalies. SIDS is defined as a NIDS in which data is categorized into known classes of intrusions. A SIDS is used most commonly for classification. A HIDS generates new frameworks through combinations of AIDS and SIDS to obtain more accurate results for the detection of both known and unknown classes of attacks.

3.2 PCA Introduction

Principal Component Analysis (PCA) is a feature extraction technique that aims to summarize the original features into principal components that represent all the features according to their importance in deciding the result. The goal of this process is to generate new and fewer features to replace the unnecessary features in the preprocessed dataset; at the same time, information loss should be minimized.

Assuming a set of n variables each containing m data in a multi-dimensional space, the covariance between any two random variables $X1$ and $X2$ is calculated by equation 1, where $X1$ and $X2$ are data in the two variables.

$$\begin{gathered}Cov(X,Y) = \frac{\Sigma(X1_i - \bar{X1})(X2_i - \bar{X2})}{m-1},\\ \bar{X1} = \frac{\Sigma(X1_i)}{m}, \bar{X2} = \frac{\Sigma(X2_i)}{m}.\end{gathered} \tag{1}$$

The covariance matrix C of each variable to another is written as the following $m \times m$ matrix, representing the distribution magnitude and direction of the features in a multivariate analysis. PCA aims to produce a diagonal covariance matrix by maximizing the variability, or “uniqueness,” of the projection of data points along each principal component to gain the most information from a limited number of principal components.

$$\begin{bmatrix} Cov(X1,X1) & Cov(X1,X2) & ... & Cov(X1,Xm) \\ Cov(X2,X1) & Cov(X2,X2) & ... & Cov(X2,Xm) \\ ... & ... & ... & ... \\ Cov(Xm,X1) & Cov(Xm,X2) & ... & Cov(Xm,Xm) \end{bmatrix}$$

Eigenvalues λ and eigenvectors $\mathbf{v}$ of each variable are calculated on the covariance square matrix by equation 2, where each λ is a constant and each $\mathbf{v}$ is a $m \times 1$ matrix.

$$C \times \mathbf{v} = \lambda \times \mathbf{v}. \tag{2}$$

The transfer matrix P is computed on the eigenvalues and eigenvectors to sum the features into principal components. The eigenvalues denote the variance of a variable to the total variance of all variables. In other words, the variance of a variable k on total population variance is caculated by equation 3.

$$\frac{\lambda_k}{\lambda_1 + \lambda_2 + ... + \lambda_m}. \tag{3}$$

The transfer matrix is generated by the ordering $|\lambda_i| > |\lambda_{i+1}| > |\lambda_{i+2}| > ... > |\lambda_m|$ and sorting the eigenvectors into a matrix $P = [\mathbf{v}_i, \mathbf{v}_{i+1}, \mathbf{v}_{i+2}, ..., \mathbf{v}_m]$.

PCA matrix multiplication based on feature importance transforms the features into a smaller number, while maintaining as much importance as they had using the above algorithm. In the following sections, principal component will be shortened as “*PC*.”

3.3 Machine Learning Algorithms

Multiple linear regression is a technique applied to continuous data such that a best-fitted linear model is generated. The most commonly used least-square regression measures the minimal sum of square distances between each data point and the predicted value. In a set of n principal components $\{PC_0, PC_1, ..., PC_{n-1}\}$, the result (0 benign or 1 abnormal) is generated through equation 4, where β is a set of coefficients for each feature, y denotes result, and e denotes the error value.

$$y = \beta_0 + \beta_1 \cdot PC_0 + \beta_2 \cdot PC_1 + ... + \beta_n \cdot PC_{n-1} + e. \tag{4}$$

A Classification and Regression Tree (CART) training algorithm is a tree-like model constructed based on values of the principal components. From the root node, each additional decision node is generated by splitting its parent node. Without hyperparameters to limit the growth of the tree, the process repeats itself until all components are used, all data are calculated, or all values of the splits are identical. Here, a classification decision tree is used based on

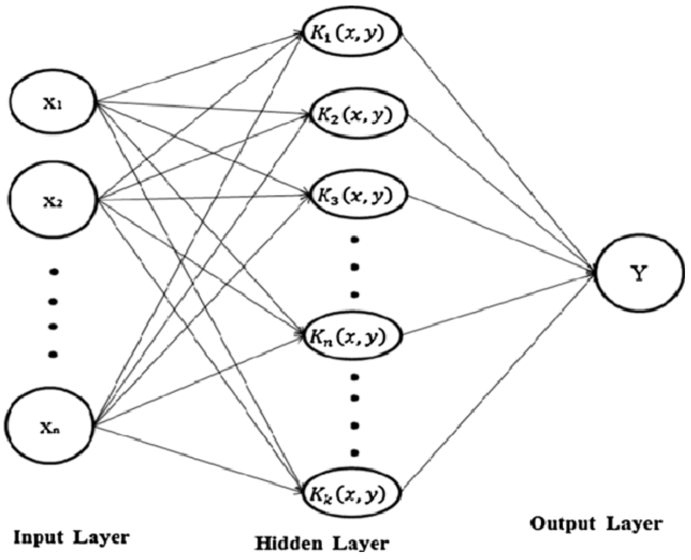


Figure 4: Illustration of layers in the RBF-kernel SVM algorithm [20].

the measurement of Gini impurity to minimize misclassification. The Gini index is the criteria to select the best principal component at each node in a classification tree, so that the probability of any data to be wrongly classified is minimized. The Gini index at each node is calculated by equation 5, where p_i represents the probability of data being classified into a particular category.

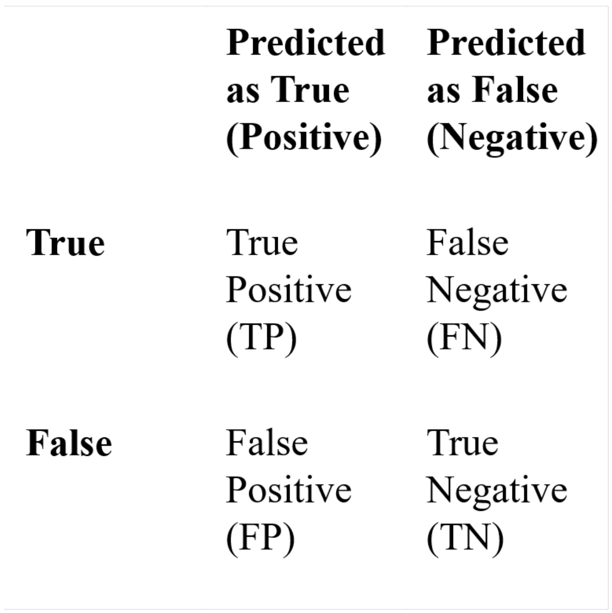
$$\begin{aligned} Gini = \sum_{i=0}^{n} p_i(1-p_i) = 1 - \sum_{i=1}^{n} (p_i)^2, \\ p_i = \frac{\text{\# data in Class i}}{\text{\# of data}}. \end{aligned} \tag{5}$$

Hence, when the Gini index is equivalent to 0, the impurity of that specific classification is also 0, and no data is wrongly classified into that category.

The support vector machine algorithm is used to classify the anomalies into different categories given in the datasets due to its feasibility in high dimensional spaces. The algorithm is based on the idea to separate the data points in a high dimensional space (data that must be modelled using more than 3 coordinates axes) into clusters based on hyperplanes. The higher dimensionality, the more accurate a result SVM outputs compared to other machine learning algorithms. The length of margins around a hyperplane and the radius of influence of each data point have significant impact on the classification of a data point into each cluster. The larger the margin, the more likely the algorithm will output a hyperplane that more accurately divides clusters. A SVM model can be generated using several different kernels, or measures of similarity between data. The radial basis function (RBF) kernel first projects data to higher dimension to determine a suitable projection of data by the functions $\phi(x)$ as shown in Figure 4. $K(x,y) =< \phi(x), \phi(y) >$, where $K(x,y)$ is the RBF kernel function. The output finds a single result by linearly summing up all results obtained in each $K(x,y)$. In RBF-kernel SVM, the kernel function between two variables x and y is expressed as equation 6, where $||x-y||^2$ is the squared Euclidean distance function, or $\Sigma(x_i - y_i)^2$, and $\gamma = \frac{1}{\sigma^2}$.

$$K(x,y) = e^{-\frac{||x-y||^2}{\sigma^2}} = e^{-\gamma||x-y||^2}. \tag{6}$$

Combinations of data from the kernel function are calculated to find the most accurate hyperplane in replacement of the projection function $\phi(x)$ for each data point. Using an RBF-kernel SVM algorithm, data are grouped and bounded by enclosed hyperplanes.



	Predicted as True (Positive)	**Predicted as False (Negative)**
True	True Positive (TP)	False Negative (FN)
False	False Positive (FP)	True Negative (TN)

Figure 5: An illustration of the confusion matrix

3.4 Measuring Metrics

This paper provides the following precision-measuring instruments for both algorithms: the accuracy score (detection rate), recall, precision, and Receiver Operating Characteristic (ROC) curves. Accuracy score, recall, and precision are all measurements of the confusion matrix in Figure 5. Accuracy score, by equation 7, is a model's ability in correctly labeling any data. A larger accuracy score indicates an overall better model. Recall is the percentage of positive data from the correctly measured data. It represents how permissive a model is to classify an instance as positive rather than negative in all correctly labeled data. The error rate is (1.00-accuracy score).

$$Accuracy\ score = \frac{|TP| + |TN|}{|TP| + |FN| + |TN| + |FP|}. \tag{7}$$

Recall is calculated by equation 8. A higher recall percentage is associated with less false negative data predicted. In the AIDS section, since this research aims to search for abnormal network traffic, positively predicted data is preferred over negatively predicted data, and higher recall rates indicate safer models.

$$Recall = \frac{|TP|}{|TP| + |FN|}. \tag{8}$$

Precision, by equation 9, is the percentage of correctly identified positive data from all predicted positive data. It measures the ability of the model to correctly identify positive data in regard to all falsely detected positive data. A higher precision suggests lower probability of wrongly labeling a normal instance. The false alarm rate is (1.00-precision).

$$Precision = \frac{|TP|}{|TP| + |FP|}. \tag{9}$$

The Receiver Operating Characteristics (ROC) curve is an instrument for measuring the probability of correct detection of anomalies in AIDS. They are graphs that relate the true positive rate (TPR) with the false positive rate (FPR) at every classification threshold between two classes: normal and abnormal. The Area Under Curve (AUC) is a representation of the probability of placing any positive data sample to the right of a negative data sample in Figure 6. In other words, it is the probability of correctly identifying any true positive data. Hence, AUC

Figure 6: Ranking the predictions from negative to positive.

analysis emphasizes the correctness of an anomaly being identified.

Specific to the linear regression algorithm, statistical variations of the model compared to the actual training data can be explained by statistical values, with the adjusted R^2 being one of the most representative measuring instruments in multiple linear regression. Adjusted R^2 is calculated based on equation 10, where R^2_{sample} denotes sample $R^2 = \frac{\Sigma(\hat{y}_i - \bar{y})^2}{(y_i - \bar{y})^2}$, $\hat{y}_i$ is the predicted value, y_i is the true value of data point, n is the total number of data, and $\bar{y}$ is the average value. It a measure of how much the dependent variable varies with the independent variables, taking the number of independent variables, or the feature-reduced principal components, into account. In other words, adjusted R^2 is the fitness of the model using multiple predictors. The value usually ranges from 0 to 1, representing the percentage of variation. The closer R^2 is to 1, the fitter the model.

$$R^2 = 1 - \frac{(1 - R^2_{sample})(n-1)}{n - 1 - \#PC}. \tag{10}$$

Residual plots and Component-Component Plus Residual (CCPR) plots are also provided as partial visualizations on the analysis of each independent variable. Since residual $(\hat{y}_i - y_i)$ measures the difference between the predicted value and the data point's actual value, if there exists an independent variable that can perfectly linearly model the results, the residual plot should resemble a horizontal line that lies on the x-axis. CCPR plots provide partial regression plots for each independent variable in relation to the results predicted solely by the variable.

In addition, the construction and fitting time of each algorithm is also considered as a measurement of their efficiency. The fitting time, or the actual detection time in practical usage of the models already generated, is a direct measurement of the algorithm's performance in real situations. The construction time is also taken into consideration, but more trivially, as a measurement of the ease of generating new models on data described by different features or data labeled by different attack classes.

4 Implementation

In this paper, Python libraries are used, including mainly *NumPy*, *Pandas*, *Scikit-Learn*, *Statsmodels*, and *Matplotlib*.

Basic libraries *NumPy* and *Pandas* are used to read and transform data into readable forms. *Pandas* reads local CSV files into the programs and converts them into data frames for further uses. While standardizing or normalizing the data frames, *NumPy* places a role of conversion from data frames into NumPy arrays due to format restrictions.

The library *Scikit-Learn* is popular among nearly all machine learning algorithms and adopts the time-efficient algorithms to solve classifier models. On the other hand, *Statsmodels* contains powerful statistical analysis instruments. In this paper, *Scikit-Learn* is used to generate the decision trees, while *Statsmodels* is used for multiple linear regression models. It is also used for standardization of data to lower variance, PCA feature extraction, and grid optimization of hyperparameters in decision trees. While many referenced papers use feature extraction, in this paper, principal component analysis is used to generate integrated components, which are used as features in later model training.

In *Scikit-Learn*, the module *model_selection* is used to split large data frames into train and test groups and apply grid optimization on input arrays for the best hyperparameters of the decision tree. The module *decomposition* is used to standardize the data in each column for more convenient PCA; the *PCA* function from *decomposition* generates a PCA object that takes in the original data frame and the number of principal components and outputs the data frame transformed to be described by the principal components. The module *metrics* imports the confusion matrix for faster accuracy score, recall, and precision calculations. The class *tree* generates a CART tree object and creates a decision tree based on the input hyperparameters. The class *svm* yields a high dimensional SVM classifier based on the hyperparameters.

Matplotlib is a library used here for plotting the results obtained from *Scikit-Learn* and *Statsmodels*, which includes heatmaps, residual and CCPR plots, ROC curves, decision tree topology, etc.

5 Experimentation

In this section, the results of the implementation of different machine learning models are demonstrated. All experimentations are run on ThinkPad X1 computer with 10^{th} Generation Intel® Core™ i7-10510U Processor (4 cores, 8 thread, 1.80GHz turbo boost, 8MB cache) [21].

5.1 HIDS Framework

As stated, the main objective of this research is to correctly identify whether a data sample is benign or belongs to an attack class known to the datasets using a machine learning approach.

The basic framework of training the hybrid IDS is provided in Figure 7. The system is composed of a feature reduction process, one AIDS component, and one SIDS component. Multiple linear regression and decision trees are both candidate algorithms for anomaly detection, while support vector machine is used for signature classification of anomalies. The reason for this choice will be discussed in the following subsections. A SIDS should be able to detect anomalies at high accuracy, recall, and low time costs. An AIDS should be able to correctly classify each type of DoS/DDoS attack for mitigation with the attacks.

We also propose simple genetic bridge between the data not categorized and the generation of a new attack class in SIDS is built. However, this paper will not elaborate on the identification of unknown data using SVM due to a lack of available resources. A simple methodology is provided for potential practical usages of this hybrid framework: to define such data as a new class of attack if and only if they are not in any data point's sphere of influence from any known clusters, and they occupy at least a certain bandwidth of data values to ensure these data are not outliers.

5.2 Datasets Description

The two different datasets used in this paper are *KDD-CUP 1999* and *CICIDS-2017*. Considering the lack of data resources, these two datasets are chosen particularly due to their comprehensive description to overall network environment and details of TCP flags during network communications. In addition, they also consist of all three key classes of DoS/DDoS as discussed in Section 2.

The *KDD-CUP 1999* dataset has been used commonly in studies relating to DoS network attacks. It simulates an attack in a military network environment. The types of attacks in *KDD-CUP 1999* can be classified into four main categories:

1. DoS (Denial of Service): The most common and classical attack that attempts to make a network service or site unavailable to legitimate users.

2. U2R (User to Root): The attacker gains access to the site by secretly stealing users' information illegally and takes advantage of a site's vulnerabilities to gain root access to the system.

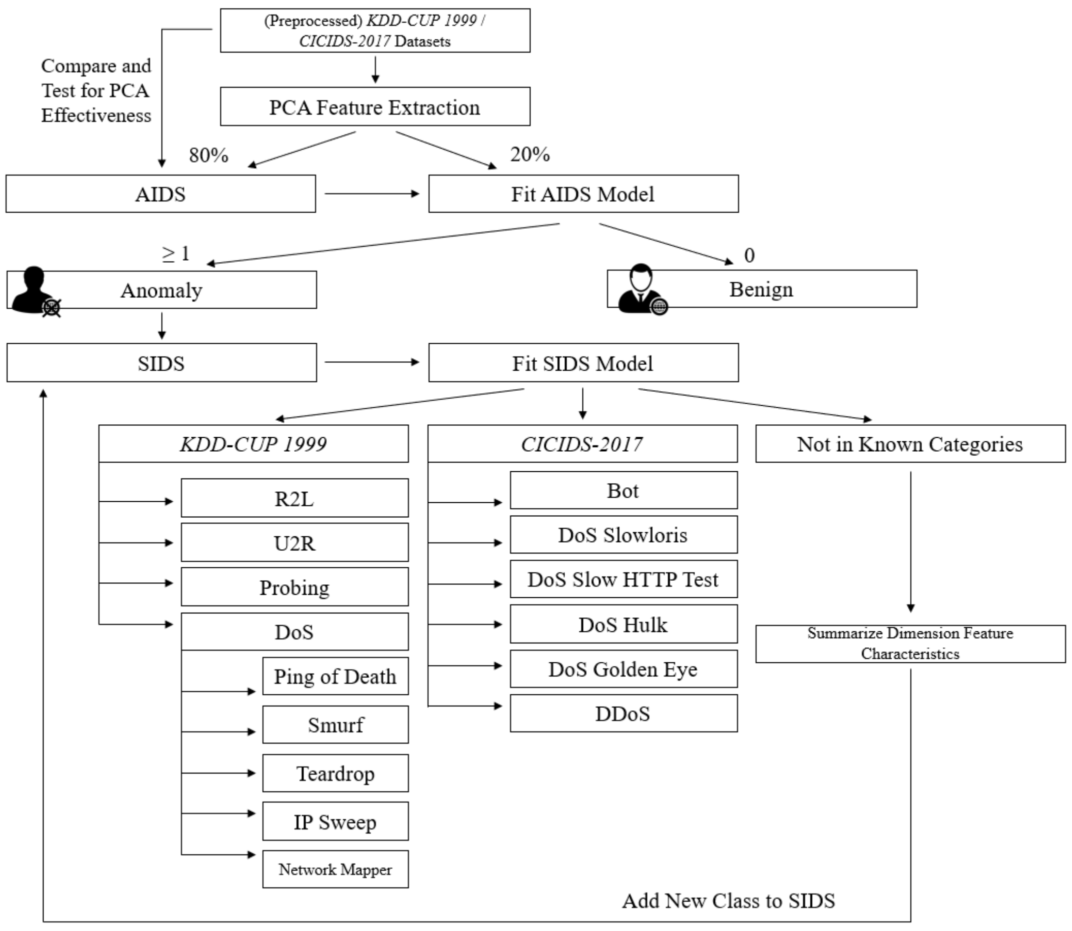


Figure 7: Flow graph for the basic framework of the proposed HIDS.

KDD-CUP 1999 Labels Categorized	
Attack Class	Label
normal	Benign back land neptune
DoS	pod smurf teardrop ipsweep nmap
Probing	portsweep satan ftp_write guess_passwd imap multihop
R2L	phf spy warezclient warezmaster buffer_overflow loadmodule
U2R	perl subtotal rootkit

Table 2: *KDD-CUP 1999* Labels Categorized [22]

3. R2L (Remote to Local): The attacker, who is capable of sending packets from a machine, exploits vulnerabilities of system to gain local control of a user.

4. Probing: The attacker illegally gains information about a network for the purpose of attacking it.

Table 2 categorizes each of the 24 data labels in *KDD-CUP 1999* into either the benign class or the four attack classes stated above.

Many referenced papers proposed models only feasibly trained and tested with 10% of *KDD-CUP 1999*, while in this paper, all data contained are used to train and test the models to simulate the big network traffic flow and overall accuracy of the models.

The *CICIDS-2017* dataset is PCAP (Certified Associate in Python Programming) qualified and contains data that resembles real-world network traffic. The dataset contains network traffic analysis using CICFlowMeter. Since *CICIDS-2017* is a package containing eight CSV files, each with different types of network attacks, a new CSV file is created, containing all data samples from the dataset that have DoS/DDoS type labels.

Table 3 contains the descriptions for each file from the downloaded *CICIDS-2017* dataset.

Table 4 shows the composition of each type of label in each dataset. It is observed that the *KDD-CUP 1999* dataset contains more than 98% of data labeled either “benign” or “DoS,” while the data used for *CICIDS-2017* training has been additionally extracted for DoS/DDoS types specifically. Both datasets are ready to be used for DoS/DDoS anomaly detection. In

CICIDS-2017 CSV files		
Time	Content	Size
Monday	Benign	176,927,918B
Tuesday	Brute force (SSH-Patator, FTP-Patator)	135,078,995B
Wednesday	DoS/DDoS (Slowloris, Slowhttptest, Hulk, GoldenEye)	225,166,395B
Thursday Morning	Web Attack (Brute force, XSS, SQL Injection)	52,023,263B
Thursday Afternoon	Infiltration (Dropbox download, Cool disk)	83,102,436B
Friday Morning	Botnet ARES	58,316,725B
Friday Afternoon 1	Port Scan	76,906,168B
Friday Afternoon 2	DDoS LOIT	77,123,859B

Table 3: Table of the descriptions in all CSV files from *CICIDS-2017* dataset

addition, there is an extremely heavy emphasis on volumetric attacks in the *KDD-CUP 1999* DoS class. However, the dataset contains the least amount of information in application-layer attacks, with only one specific type of attack method given. On the other hand, the *CICIDS-2017* dataset favors application layer attacks. When both are observed and experimented with, they cover all DoS/DDoS attack classes.

The sizes of the datasets are stated in the following:

$$\textit{KDD-CUP 1999} = [4898431 \times 42] \text{ (683,596 kB)}$$
$$\textit{CICIDS-2017} = [1109470 \times 78] \text{ (367,582 kB)}$$

Each dataset is divided into 80% of training data and 20% of test data randomly.

We draw down similarities between the features of the two datasets. Below are how *KDD-CUP 1999*'s features are categorized.

1. Features of Individual TCP Connections;
2. Features of Connections in Domain Knowledge;
3. Traffic Features Captured in Two Seconds.

Class 1 contains features that are recorded in the TCP layer, including the duration of the connection, the protocol type, the lengths in bytes of a packet, and the status of the packet. Class 2 mainly contains information regarding the number of wrong fragments, number of failed login attempts, and number of outbound FTP sessions. Class 3 contains the percentage error rates by "SYN" or "REJ" errors. Therefore, features in Class 3 are all continuous instead of discrete (categorical). The *CICIDS-2017* dataset contains features that mainly summarize aspects of the packet. These include time stamp, source, destination IPs, source and destination ports, packet length, and TCP flags counts ("FIN", "SYN", "RST", "PSH", "ACK", "URG", "CWE", "ECE").

Although there exists a shortcoming that the two datasets do not share a large number of common features, similarities between them describe the importance of two major fields in network anomaly detection:

1. Total duration of the data packet;
2. Total number of TCP flags and related error rates.

Thus, we hypothesize that these two fields of features are dominant in the identification of anomaly connections.

KDD-CUP 1999	
Label	Size
Benign	64.84%
DoS	33.38%
Probing	1.71%
R2L	0.07%
U2R	3.42%
KDD-CUP 1999 DoS	
pod (Protocol)	0.02%
teardrop (Protocol)	0.05%
smurf (Volumetric)	99.43%
ipsweep (Volumetric)	0.44%
nmap (Application)	0.06%

CICIDS-2017	
Label	Size
BENIGN	65.51%
Bot (Application)	0.18%
DDoS (General DDoS)	11.54%
DoS GoldenEye (Application)	0.93%
DoS Hulk (Application)	20.83%
DoS Slowhttptest (Application)	0.50%
DoS slowloris (Application)	0.52%

Table 4: Table of the composition of each label in the datasets

5.3 Preprocessing the Datasets

The transformation of the datasets into formatted data frames is an indispensable process. According to the label of each data sample, a binary representation of whether the data is clean or not is generated to replace the original label for the convenience of algorithm-training. Since the datasets contain labels with different types of DoS/DDoS attacks, during anomaly detection, the labels are first converted to numerical values. The values are first assigned to be 0, denoting normal connections, and 1, denoting anomalies, to make the training process for AIDS more convenient. The values are stored in a new column, named "result." In the SIDS section, the abnormal data are then classified into each type of DoS/DDoS attack depending on the datasets, labeled with an integer ≥ 1. Data containing empty values in *CICIDS-2017* are cleaned to ensure only functioning data are kept.

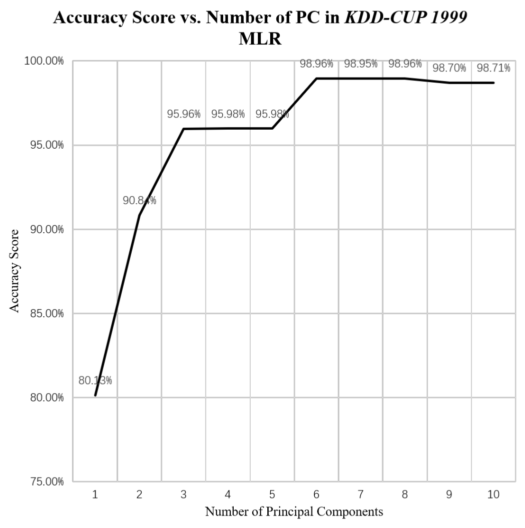
After labelling the data frames, categorical columns in the datasets must also be processed to be trained. We separate each value in a categorical feature into different columns to consider them as numerical values independently during PCA. In each of these generated columns, 0 is used to represent the datum that do not contain the value, and 1 is used to represent those containing said value in its original column.

5.4 PCA Feature Extraction

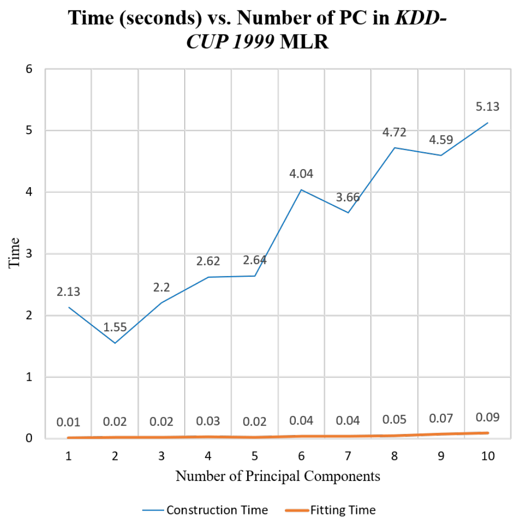
The technique of PCA feature extraction is applied to the datasets through matrix multiplication. When the transfer matrix P is applied on a dataset S, the result matrix R is the PCA feature extracted data in the form of a matrix.

$$R\ [\#\text{Data} \times \#PC] = S\ [\#\text{Data} \times \#\text{Features}] \cdot P^T\ [\#\text{Features} \times \#PC]$$

The principal components decrease in importance. In other words, PC_0 is a best fitted version of all features and contributes the most variance. The more number of principal components are generated, the less is explained variance ratio of a later generated principal component due to



(a) Accuracy Score



(b) Construction and Fitting Time

Figure 8: Line charts of how the multiple linear regression model accuracy score and time varies with the number of principal components in *KDD-CUP 1999*.

its decreased importance.

In each dataset, and in each training model selected, the best number of principal components varies depending on the accuracy of the result.

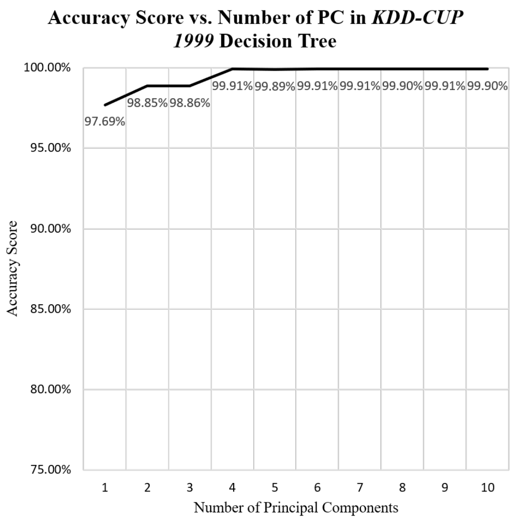
It is simple to identify a threshold in the multiple linear regression models for *KDD-CUP 1999* dataset when visualization is presented in Figure 8(a). When the number of principal components is increased to 6, the accuracy score is significantly increased. When the number of principal components exceeds 6, the accuracy scores remain relatively constant. To maintain a low construction and fitting time according to Figure 8(b), it is reasonable to select a number of 6 principal components in the multiple linear regression algorithm for training the *KDD-CUP 1999* dataset.

As Figure 9(a) shows, in the case of a decision tree, the *KDD-CUP 1999* dataset demonstrates stability when the number of principal components increases to 4, with an accuracy score as high as 99.91%. The accuracy score of the decision tree model starts to decrease after reaching its maximum threshold. Such a phenomenon is due to unnecessary new components that drive the model farther away from the best-fitted component. At the same time, low fitting time is maintained, as shown in Figure 9(b).

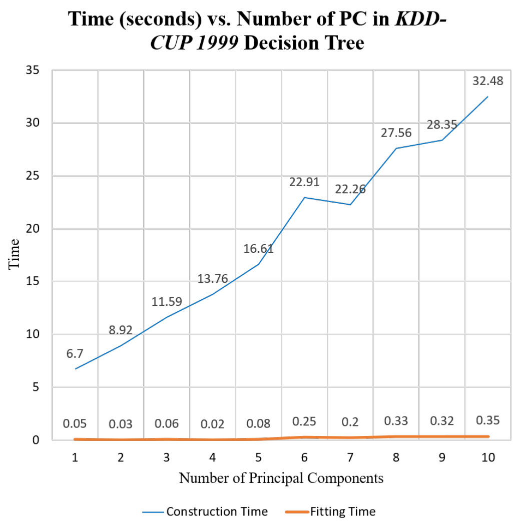
Figure 10 is a heatmap representation of the importance of original features, with brightness or darkness of a feature indicating the significance of its variance, or influence, it has. It can be inferred from PC_0 that the series of features “serror_rate” and “rerror_rate” (the rate in which SYN and REJ error flags appear) are generally the most important, while “protocol_type” also exhibits its impact on the overall distribution of both PC_0 and PC_1.

As shown in Figure 11(a), in the multiple linear regression model for *CICIDS-2017* dataset, the maximum threshold of accuracy score is reached after the number of principal components generated is leveled to 7. After considering the reasonable increase in time in Figure 11(b), a number of 7 components is selected for this multiple linear regression training model.

Using a decision tree model, it can be inferred from Figure 12(a) and (b) that 5 principal



(a) Accuracy Score



(b) Construction and Fitting Time

Figure 9: Line charts of how the decision tree model accuracy score and time varies with the number of principal components in *KDD-CUP 1999*.

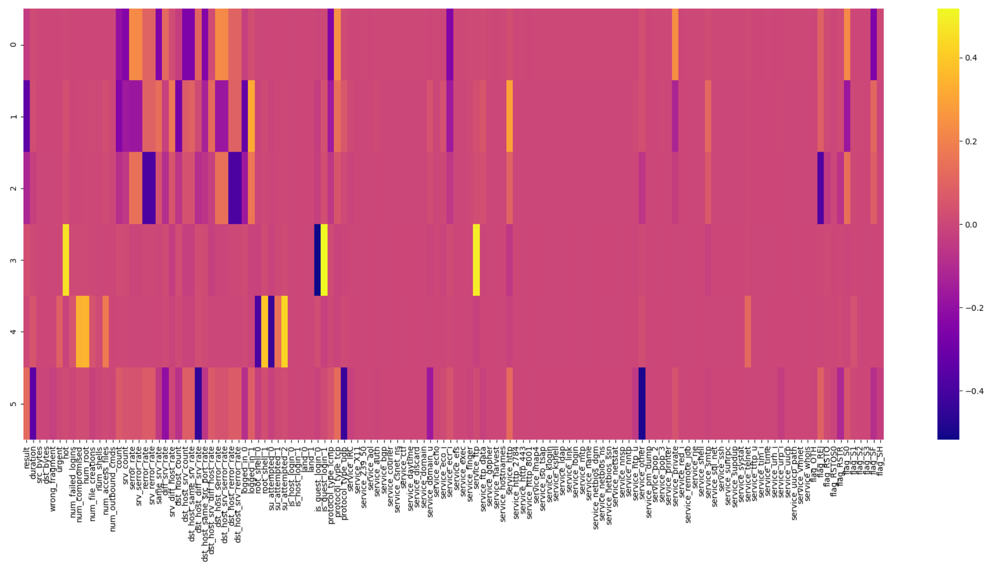
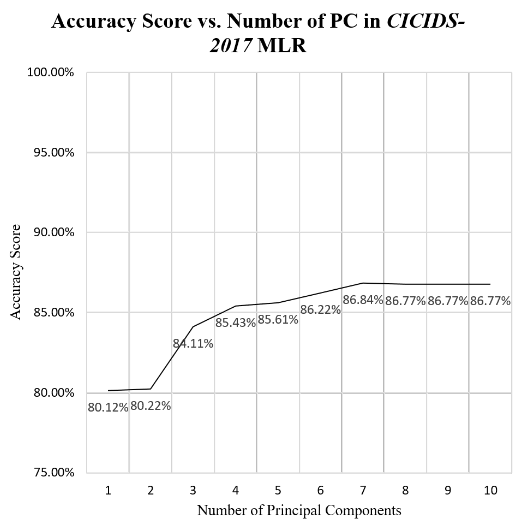
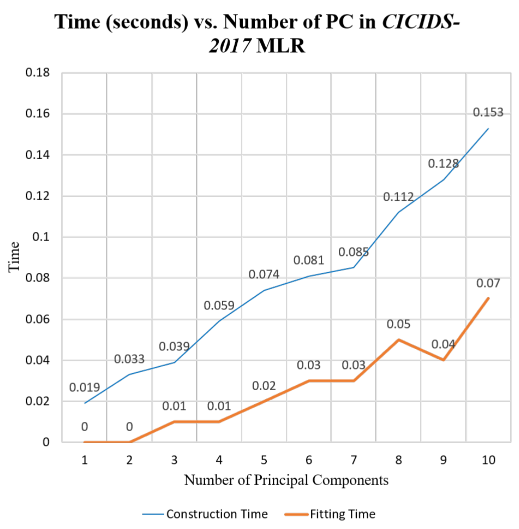


Figure 10: Heatmap for the importance of different features in the *KDD-CUP 1999* dataset that are used to generate the six new features, denoted by their indexes as 0, 1, 2, 3, 4, and 5, respectively.

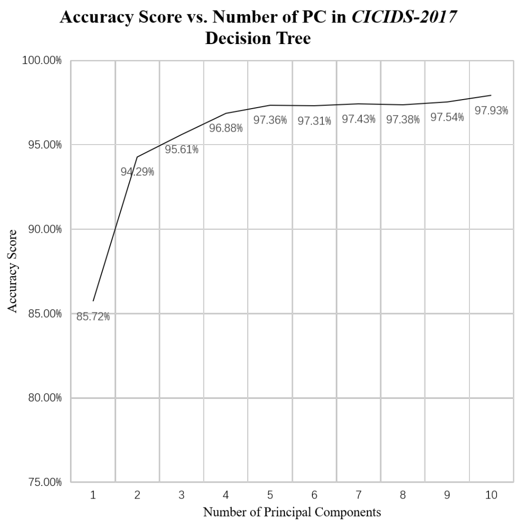


(a) Accuracy Score

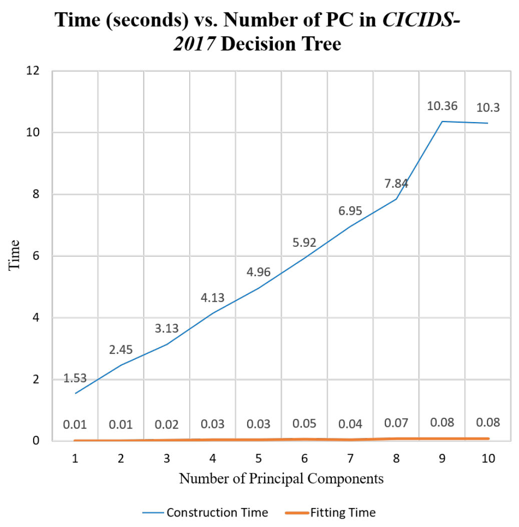


(b) Construction and Fitting Time

Figure 11: Line charts of how the multiple linear regression model accuracy score and time varies with the number of principal components in *CICIDS-2017*.



(a) Accuracy Score



(b) Construction and Fitting Time

Figure 12: Line charts of how the decision tree model accuracy score and time varies with the number of principal components in *CICIDS-2017*.

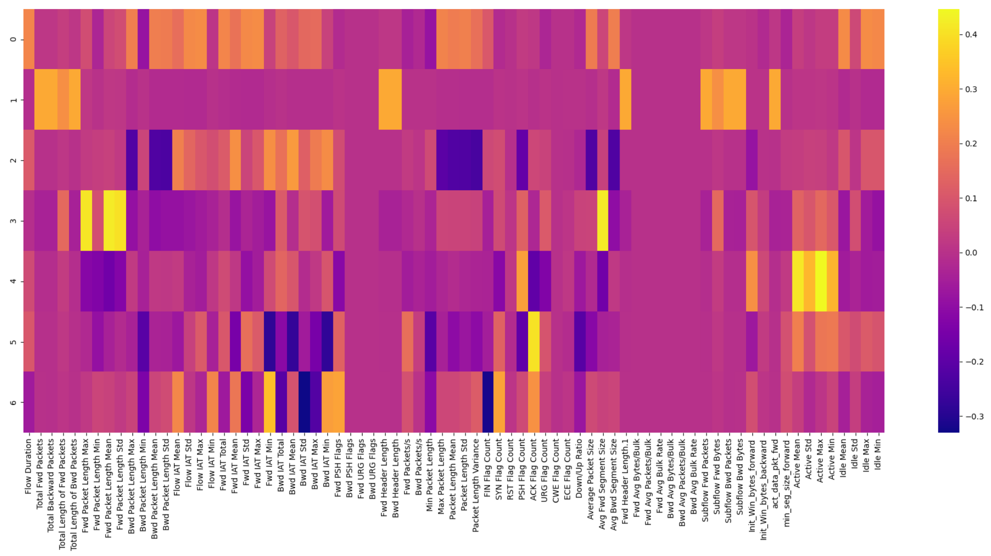


Figure 13: Heatmap for the importance of different features in the *CICIDS-2017* dataset that are used to generate the seven new features, denoted by their indexes as 0, 1, 2, 3, 4, 5, and 6, respectively

components are needed to generate an accurate decision tree with a score of 97.36% and a fitting time of only 0.03 seconds.

In Figure 13, it is easy to identify features that are especially brighter or darker. In both PC_0 and PC_2, the IAT (Inter Arrival Time) length and status, packet length, and idle time and status are highlighted. In PC_1, other features are identified, including the total number of forward and backward packets and the size of forward and backward sub-flow packets.

The results mostly correspond to the hypothesis proposed in Subsection 5.2 by comparing important features contained in both datasets. Protocol types and "SYN"/"REJ" error flags both play a major role in determining the principal components in *KDD-CUP 1999* and *CICIDS-2017*. However, counter-intuitively, the duration of the connection does not have a significant impact on the result. Replacing the total duration of time is the feature of "flow IAT" in *CICIDS-2017*. Since attackers often spoof their IP addresses while DDoS attacking servers, the total time length of their connections will be renewed, and then recounted from the start of the new connection from their spoofed addresses. In replacement of the total time length, the flow Inter Access Time, the connection time intervals between each entry into a single system, obtains higher influence in determining the legality of network traffic. IAT is delayed due to the lack of ability to handily deal with packet floods.

Using PCA, the time needed to run each model is reduced significantly compared to simply using all features to generate the prediction models without any processing. At the same time, a relatively constant accuracy score is maintained, as shown in Table 5. In addition, PCA also resolves the issue of over-fitting the multiple linear regression models, which will be elaborated in the subsection 5.5.1.

5.5 AIDS

We introduced and compared two supervised machine learning algorithms to decide the model for the AIDS component: multiple linear regression and decision tree. These two algorithms selected as AIDS candidates due to their low computational complexities and high accuracy

Use of PCA	Total Computation Time	Accuracy score
KDD-CUP 1999 / MLR		
Without PCA	66.69 seconds	99.60%
With PCA	4.04 seconds	98.96%
% Increase	-93.94%	-0.64%
KDD-CUP 1999 / Decision Tree		
Without PCA	133.86 seconds	99.84%
With PCA	13.78 seconds	99.91%
% Increase	-89.71%	0.07%
CICIDS-2017 / MLR		
Without PCA	2.35 seconds	92.94%
With PCA	0.09 seconds	86.84%
% Increase	-96.38%	-6.56%
CICIDS-2017 / Decision Tree		
Without PCA	10.67 seconds	98.25%
With PCA	4.99 seconds	97.36%
% Increase	-53.23%	-0.91%

Table 5: Table of comparisons between total computational time and accuracy score in different training algorithms (% Increase denotes % increase from "Without PCA" to "With PCA")

Dataset	Algorithm	Accuracy Score
KDD-CUP 1999	MLR	85.64%
KDD-CUP 1999	Decision Tree	83.54%
CICIDS-2017	MLR	72.25%
CICIDS-2017	Decision Tree	82.99%

Table 6: Table presenting the accuracy scores of MLR and Decision Tree algorithms on doing Signature-based work

scores when the job of anomaly classification is put on them. As shown in Table 6, both algorithms performed poorly on directly classifying the data into different types of attacks. The lower accuracy scores obtained by the decision tree algorithm when performing signature classification is due to the weakness that CART trees are prone to errors in different attack classes caused by an inequity in the amount of data in each category. Multiple linear regression is also quite inflexible when attempting to separate the results obtained into multiple classes only in a linear way. In addition, considering conclusions drawn from previous researches on decision trees' accuracy and efficiency in anomaly detection an a lack of experimentation to prove the multiple linear regression algorithm's effectiveness or ineffectiveness using other datasets despite [2]'s work, we provide a response to compare the classification decision trees and the multiple linear regression algorithm by conducting experimentations with the aid of PCA.

5.5.1 Multiple Linear Regression

In the prevention of overfitting multiple linear regression models, PCA also plays an important role. Overfitting often leads to unjustified high value of R^2 with over-reliance on the given training data, while unnecessary overuse of variables is the most direct cause. The most efficient method to limit the input variable is to set the number of principal components according to the Figure 8 and Figure 11, again proving the vitality of feature reduction techniques on the use of linear regression algorithms.

Figure 14 illustrates the residual plots for each principal component in the model for *KDD-CUP 1999* dataset. The residual plots for PC_0, PC_1, and PC_2 resemble linear lines that lie on the x-axis, with only slight deviation above and below the line. The residual plots for PC_3, PC_4, and PC_5 are not as evenly distributed across the x-axis as the first three, but

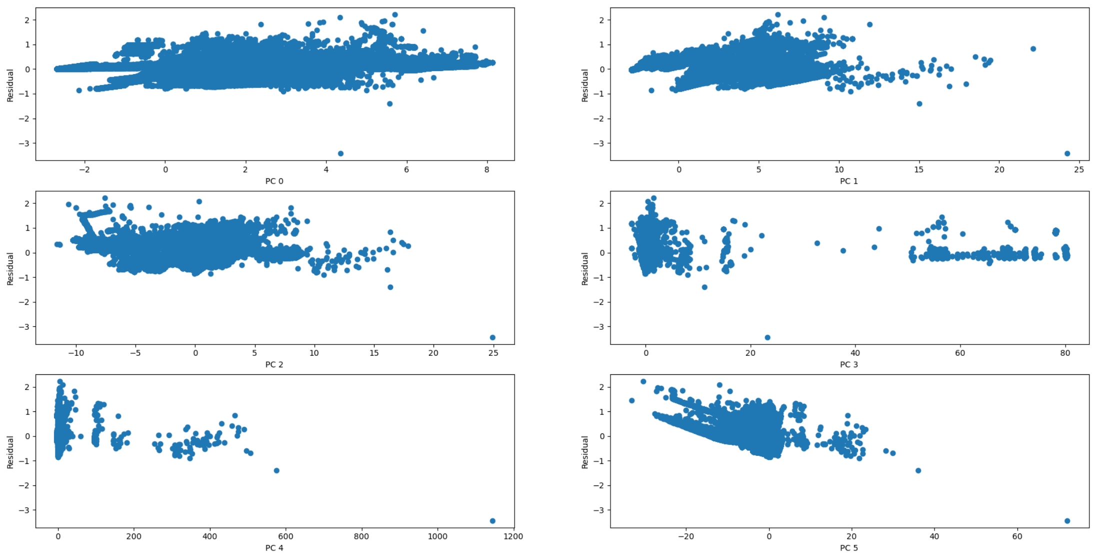


Figure 14: residual plot for all principal components in *KDD-CUP 1999*.

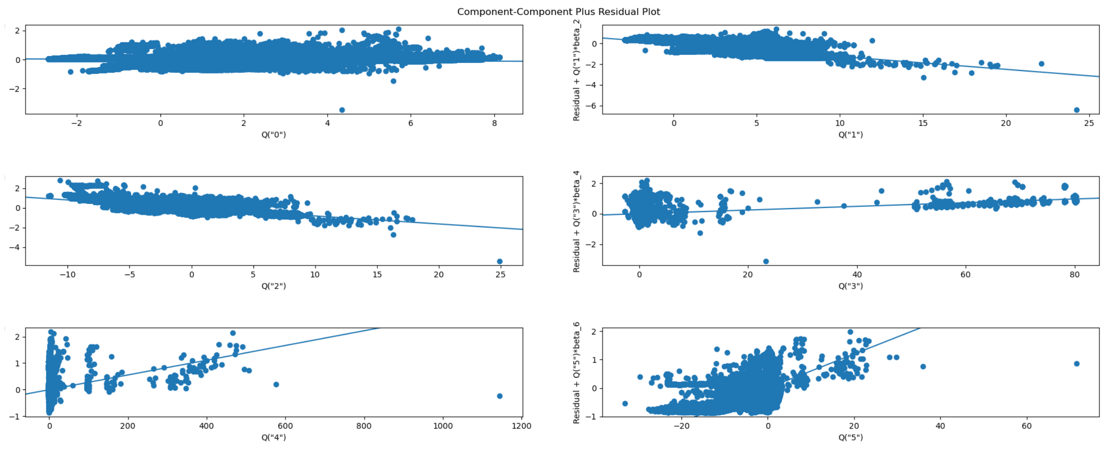


Figure 15: CCPR plots for all principal components in *KDD-CUP 1999*.

Attributes to the model	Value
R^2	0.908
Adjusted R^2	0.908
Number of observations (Rows)	3918744
Number of Residuals	3918737
Number of predictors	6
Log-Likelihood	2706900

Table 7: General summary of the 6-dimensional regression model for *KDD-CUP 1999* training data

	coefficient	Standard Error	t value	P value for $P > \|t\|$
Coefficient	$\beta_0 = 0.8014$	6.13×10^-5	13100	0.000
PC_0	$\beta_1 = -0.0134$	1.72×10^-5	-782.057	0.000
PC_1	$\beta_2 = -0.1252$	2.46×10^-5	-5082.431	0.000
PC_2	$\beta_3 = -0.0816$	2.69×10^-5	-3036.243	0.000
PC_3	$\beta_4 = 0.0121$	3.25×10^-5	371.849	0.000
PC_4	$\beta_5 = 0.0028$	3.33×10^-5	83.079	0.000
PC_5	$\beta_6 = 0.0598$	3.66×10^-5	1635.145	0.000

Table 8: Table of detailed analysis of the coefficient and principal components in the regression model for *KDD-CUP 1999*

also demonstrates low residuals. The decrease in symmetry across the x-axis as the ordered numbering of principal component rises corresponds to the decrease in PCA's ability to fit data into principal components by their features.

By the coefficients in Table 8, the prediction model for *KDD-CUP 1999* can be written as:

$$y = 0.8014 - 0.0134 \cdot PC_0 - 0.1252 \cdot PC_1 - 0.0816 \cdot PC_2 + 0.0121 \cdot PC_3 + 0.0028 PC_4 + 0.0598 \cdot PC_5 \pm 2.3240 \times 10^{-4}$$

The coefficients of the six principal components match the conclusion drawn from the CCPR plots.

As Table 7 shows, *KDD-CUP 1999*'s model obtained $R^2 = 0.908$ after PCA feature extraction. Although $R^2 = 0.983$ when all 42 features are used, the accuracy score is almost unchanged compared with the feature-reduced model. A plausible explanation for this phenomenon is that the feature-reduced model contains only 6 predictors, so each dimension varies less with the variation of other dimensions. On the other side, in a 42-dimension model, the variance of each variable closely explains another, leading to an unusually high R^2 that does not better predict the test data. The model's log-likelihood is as high as 2706900, and the P value, the probability of obtaining any prediction more extreme than the results observed, for each principal component is nearly equivalent to 0. *KDD-CUP 1999*'s model receives an accuracy score of 98.96%. At the same time, the model's recall is 98.35% and precision is 99.63%, indicating successful detection of anomalies and low false alarm rates. Figure 19 also observes that *KDD-CUP 1999*'s model obtains an AUC of 1.00, performing nicely on identifying network anomalies.

Figure 17 is the residual plots for principal components in *CICIDS-2017* training data. When compared with the residual plots for the *KDD-CUP 1999* model, the residual plots shown in the figure demonstrate less symmetry, and are all slightly tilted to the upper half of the x-axis, with PC_1 and PC_3 being two of the least symmetric. The CCPR plots in Figure 18 also demonstrate significant tilting of the regression lines for each principal components due to separated data points far away from the main cluster, which is especially true in PC_0 and PC_1's analysis. The model exhibits huge sensitivity to few pieces of data unrelated to the majority data points.

By Table 10, the prediction model can be written as:

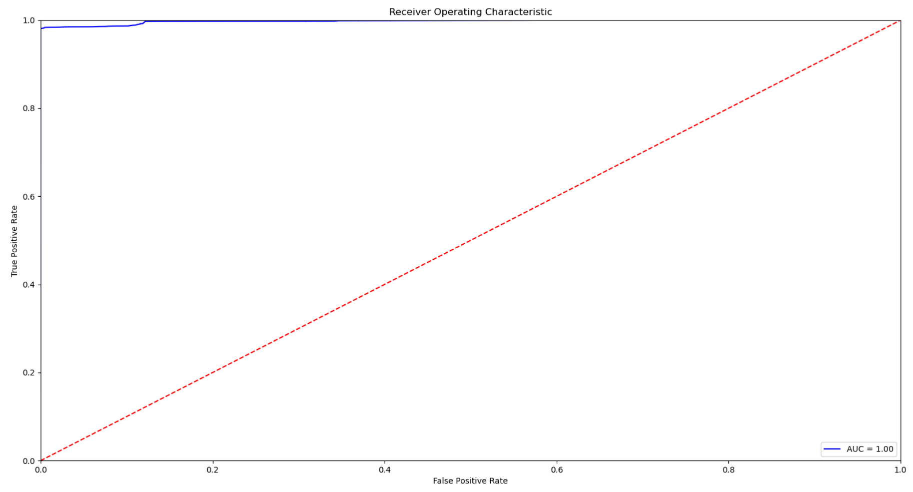


Figure 16: ROC curve for *KDD-CUP 1999* multiple linear regression algorithm

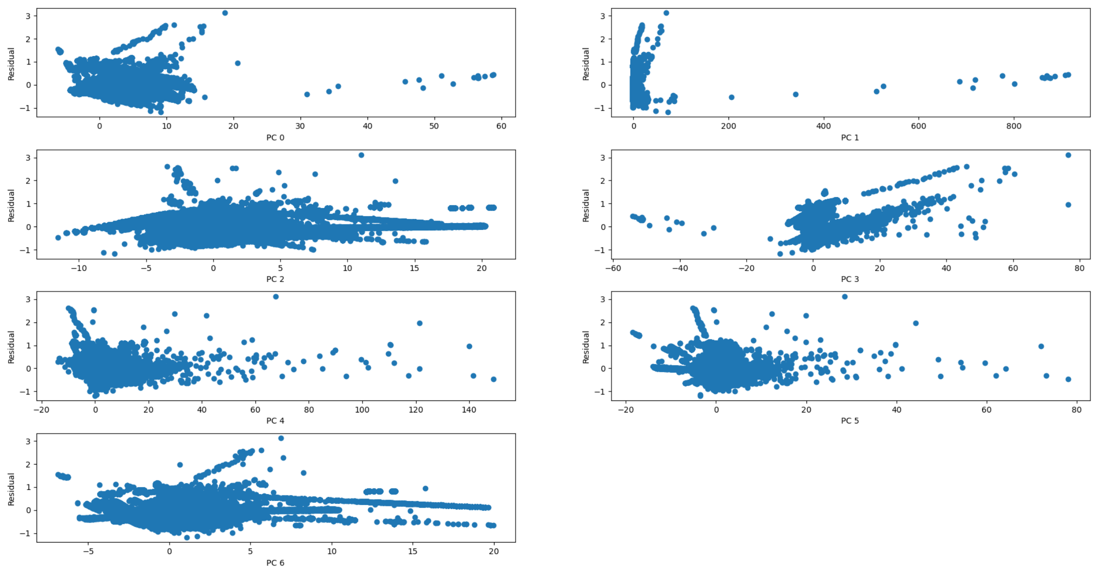


Figure 17: residual plots for all principal components in *CICIDS-2017*.

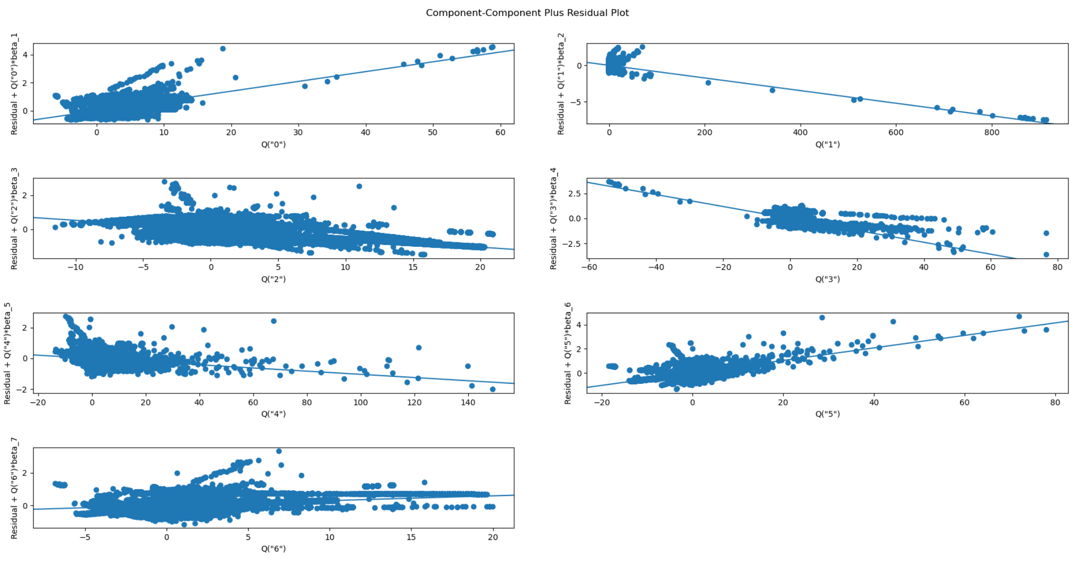


Figure 18: CCPR plots for all principal components in *CICIDS-2017*.

Attributes to the model	Value
R^2	0.522
Adjusted R^2	0.522
Number of observations (Rows)	887576
Number of Residuals	887567
Number of predictors	7
Log-Likelihood	-272730

Table 9: General summary of the 7-dimensional regression model for *CICIDS-2017* training data

$$y = 0.3445 + 0.0696 \cdot PC_0 - 0.0086 \cdot PC_1 - 0.0524 \cdot PC_2 - 0.0598 \cdot PC_3 - 0.0103 \cdot PC_4 - 0.0518 \cdot PC_5 + 0.0292 \cdot PC_6 \pm 9.5300 \times 10^{-5}$$

The multiple linear regression algorithm for *CICIDS-2017* is not as accurate as the former. *CICIDS-2017*'s model obtains an accuracy score of 86.84%. The model's recall is only 64.02% and precision is 89.64%. Statistically, Table 9 also demonstrates poor correlation between the training data and the values obtained from the equation, with an R^2 value of 0.522 and a log-likelihood of -272730. Even before PCA feature extraction to 7 principal components, the multiple linear regression algorithm only obtained a R^2 of only 0.706, since an increase in independent variables will always lead to an increase in R^2. As Figure 19 shows, *CICIDS-2017*'s model obtains an AUC of only 0.89, significantly lower when compared to the former.

Table 11 provides a summary for both the construction time, fitting time, and accuracy-measuring-metrics for both multiple linear regression models.

5.5.2 Decision Tree

In the decision tree approach, a Classification and Regression Tree (CART) structure is generated based on the principal components given. By splitting the values of principal components, a decision tree judges to which branch a given data sample belongs and returns the predicted value of its label when the last leaf is visited.

In this research, the decision trees consider three hyperparameters: the maximum depth of the tree (max_depth), the minimum number of samples needed to split the root node or any

	coefficient	Standard Error	t value	P value for $P > \|t\|$
Coefficient	$\beta_0 = 0.3445$	0.001	921.748	0.000
PC_0	$\beta_1 = 0.0696$	9.53×10^-5	730.415	0.000
PC_1	$\beta_2 = -0.0086$	0.000	-77.352	0.000
PC_2	$\beta_3 = -0.0524$	0.000	-338.233	0.000
PC_3	$\beta_4 = -0.0598$	0.000	365.053	0.000
PC_4	$\beta_5 = -0.0103$	0.000	-48.591	0.000
PC_5	$\beta_6 = -0.0518$	0.000	232.863	0.000
PC_6	$\beta_7 = 0.0292$	0.000	123.617	0.000

Table 10: Table of detailed analysis of the coefficient and principal components in the regression model for *CICIDS-2017*

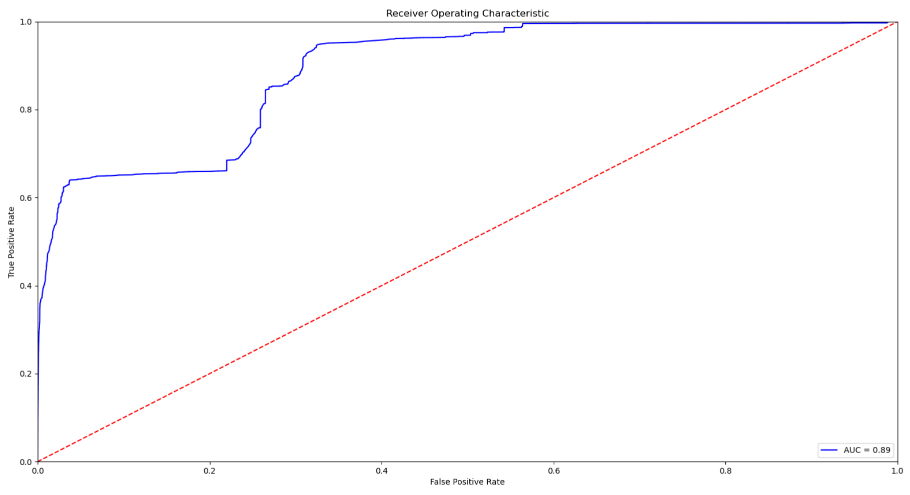


Figure 19: ROC curve for *CICIDS-2017* multiple linear regression algorithm

Model	Construction Time	Fitting Time	Accuracy Score	AUC	R^2
KDD-CUP 1999	4.04 sec	0.02 sec	98.96%	1.00	0.908
CICIDS-2017	0.09 sec	0.03 sec	86.84%	0.89	0.706

Table 11: Table of summary for the two multiple linear regression model

	Construction Time	Fitting Time	Accuracy Score	Number of Nodes	Number of Edges
KDD-CUP 1999 with PCA	13.76 sec	0.02 sec	99.91%	103	100
KDD-CUP 1999 without PCA	132.40 sec	1.46 sec	99.84%	67	64
CICIDS-2017 with PCA	4.96 sec	0.03 sec	97.36%	771	768
CICIDS-2017 without PCA	10.52 sec	0.15 sec	98.25%	71	68

Table 12: Table of the accuracy scores and time for each decision tree with and without PCA

child node (min_sample_split), and the minimum number of samples needed to generate some number of leaves (min_sample_leaf). In order to find the combination of the three parameters that can generate the maximized accuracy, every combination of the three parameters should be considered. Hence, in the creation of the hyperparameters, *GridSearchCV* from the package *model_selection*, as introduced in Section 4, is used to loop through 3 predefined arrays containing plausible values of each parameter: max_depth = $[2, 3, 4, 5, 6]$, min_sample_split = $[2, 4, 6, 8]$, and min_sample_leaf = $[2, 4, 6, 8, 10]$. These arrays guarantee a total of 100 combinations, sufficient to optimize the values of hyperparameters.

The issue of over-fitting is also considered while selecting values for the arrays. A decision tree with a maximum depth of 6 levels for the datasets is already a sufficiently accurate tree model with an acceptable amount of time. Larger depths of decision trees cause unpredictable fluctuations in accuracy score, which is especially true for the *CICIDS-2017* dataset. These fluctuations are attributed to the dataset's feature sensitivities. Occasionally, it takes an exceptional amount of time to construct or fit the model. The minimum number of samples required to split any node or decide whether to generate any leaf restricts the growth of the tree model by eliminating noises that cause bogus or unnecessary branches or leaves to be generated.

KDD-CUP 1999's tree: max_depth = 6, min_sample_split = 8, min_sample_leaf = 2.
CICIDS-2017's tree: max_depth = 6, min_sample_split = 2, min_sample_leaf = 10.

As a result, both trees received high accuracy scores, recall, and precision. *KDD-CUP 1999*'s decision tree, with an accuracy score of 99.91%, achieved a recall of 99.83% and a precision of 99.83%. From Figure 20, it can be inferred that the probability of correctly labeling any abnormal data is as high as approximately 100%.

CICIDS-2017's decision tree also obtained a high accuracy score of 97.36%, a recall of 97.44% and a precision of 95.05%. It demonstrates excellent predictability, as seen in Figure 21, with an AUC of 0.97.

As summarized in Table 12, the decision trees received efficient construction and model fitting time and high accuracy scores after PCA. Counter-intuitively, the number of decision nodes after PCA increased dramatically compared with the trees without PCA, varying inversely with time. Although the number of principal components is significantly less than the number of features, the number of splits required to output results with low Gini indexes increases. However, since the number of features without PCA that the program needs to calculate for the minimization of Gini impurities at each split is much higher than that after PCA feature extraction, and the data of features before standardization have much greater values, time of computation after PCA is significantly lower than that before.

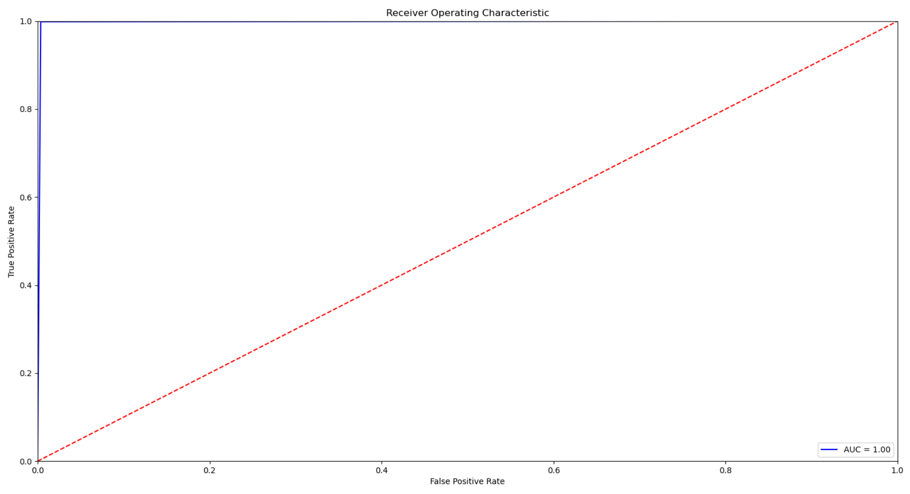


Figure 20: ROC curve for *KDD-CUP 1999* decision tree algorithm

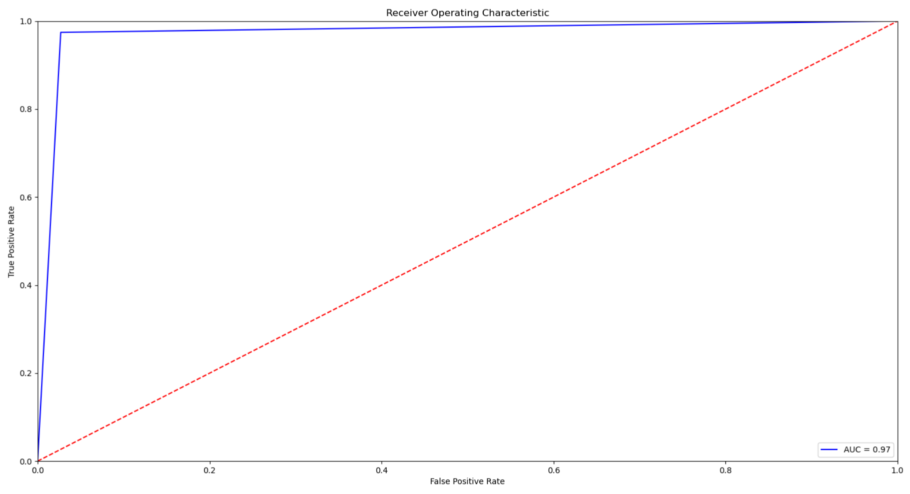
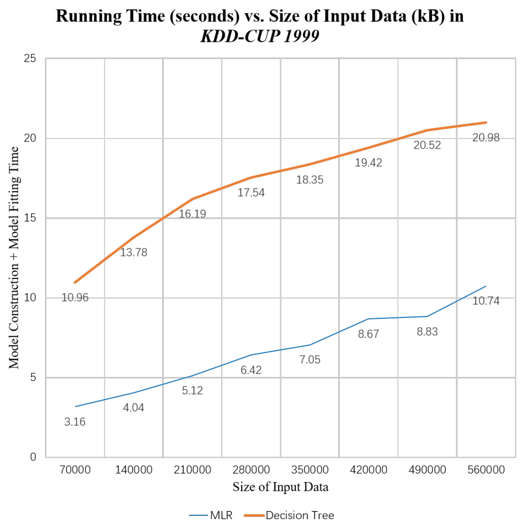
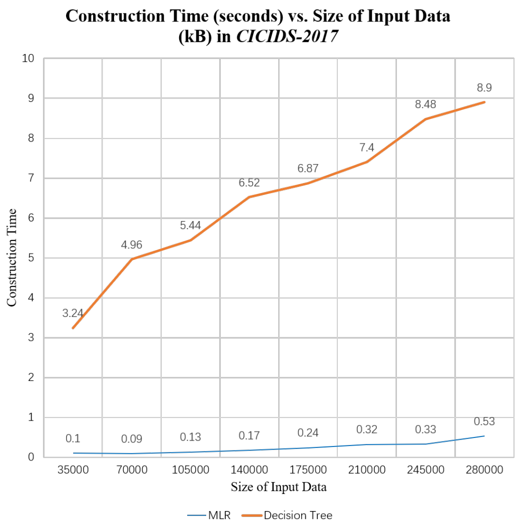


Figure 21: ROC curve for *CICIDS-2017* decision tree algorithm



(a) *KDD-CUP 1999*



(b) *CICIDS-2017*

Figure 22: Line charts of how the construction and fitting time varies with the size of input data in each algorithm

5.5.3 Result and Performance Analysis

Decision trees generally take a longer time than multiple linear regression models, but they also return a higher and stabler accuracy score. Due to the fact that multiple linear regression is too sensitive to outliers or certain correlations between some principal components, a few data samples from the randomized training data that are far away from the main cluster of data will cause the generation of a biased model. As observed, some of *CICIDS-2017*'s CCPR graphs 18 for each principal component shows unnatural clustering or tilting due to outliers far away from the main cluster. The observed pattern is that the main cluster of data points for important principal components is relatively short in spread, so distant outliers have even greater affect on the model, tilting the regression line. Henceforth, the multiple linear regression algorithm yields overly unstable results. On the contrary, a decision tree is constructed as a precise classification structure that can better predict categorical labels than regression-based algorithms, with less principal components needed, as observed in the experimentation.

Although the accuracy scores are much better, the decision tree algorithm still takes longer time to yield a model, as observed from Figure 22. Even so that is true, the increase in construction time for decision trees is not a fatal flaw. Since the construction computational complexity of a tree is $O(n_{samples} \times n_{PC} \times n_{PC})$ [23], where $n_{samples}$ is the number of data samples and n_{PC} is the number of principal components, compared with $O(n_{samples} \times n_{PC}^2)$ [24] complexity for least-square regressions, and the number of principal components needed to generate an accurate tree is observed to be less than that for a regression model, the running time of decision trees will outperform that of regression-based models as the number of must principal components increases. The time complexities of the algorithms hypothesize that when other credible datasets or captured traffic data are trained on the same algorithm, their decision trees' construction times can result in a lower value than their multiple linear regression models. In addition, the fitting time in actual use of the models are relatively constant. In fact, the decision tree algorithm maintains a lower computational complexity of $O(log(n_{samples}))$ compared to $O(n_{samples})$ in multiple linear regression. Thus, decision trees are selected as the operator for AIDS.

Table 13compares the accuracy scores with referenced literature with regard to the differences in data preprocessing and feature reduction techniques. As the experimentation in earlier sections of this paper shows, the use of PCA only causes minor deviation in accuracy scores, so differences in accuracy scores between this research, [3], [5], and [16]'s decision trees can be accounted mainly to the use of the bandwidth of data from *KDD-CUP 1999* and the algorithm's usage for detection or directly for classification. The research done in [3] and [5] yields similar results to that in this paper, but the use of a smaller amount of data to train the model and the lack of feature reduction techniques causes the generation of slightly lower result. In fact, the accuracy score without PCA feature extraction for *KDD-CUP 1999* decision tree, 99.84%, fits exactly into [5]'s range of accuracy scores. In Devi's [16] experimentation, the decision tree algorithm is directly used for classification of each attack class, leading to significantly lower accuracy than the use of decision tree for anomaly detection only; the different passed in hyperparameters to measure impurity and to restrict the growth of the tree also leads to minor changes in accuracy score. Similarly, in [13]'s decision tree trained on *CICIDS-2017*, differences in hyperparameters' values and the use of PCA feature extraction accounts for the differences.

The multiple linear regression model for *CICIDS-2017*, in comparison with Sambangi and Gondi's [2] work, which trained the model based on the Friday CSV files, provides a model on all anomaly data related to DoS/DDoS combined. Specifically, this model proved a lower accuracy score on the Friday Morning log file, but a better accuracy score on Friday Afternoon data. A 11.02% decrease in accuracy score is seen from their Information Gain (IG) feature selected model for Friday morning data of 97.86%, while a 13.05% increase is seen from their 73.79% model on Friday afternoon. While the same multiple linear regression technique is applied to train the prediction model, it can be concluded that the difference in accuracy scores can be attributed to the following two differences:

1. The use of PCA feature extraction, in place of IG feature selection;
2. The use of all data samples related to DoS/DDoS from *CICIDS-2017*, yielding a model to which samples with certain labels are more or less sensitive.

5.6 SIDS

From [15], it is justified that the SVM algorithm performs less accurately at anomaly detection than tree and forest structures. The SVM algorithm is known for its high accuracy in classifying high-dimensional data, but also its tendency to yield inaccurate hyperplanes if a small number of clusters of data distant to the many others is present. Thus, we do not directly apply the SVM algorithm on the separation of abnormal data from normal data, since data from the benign class is described by largely different values in certain important features compared to any of the abnormal classes, but instead on the classification of abnormal data into attack classes. This subsection introduces RBF kernel SVM algorithm.

5.6.1 Support Vector Machine

In the SVM-based SIDS, data predicted as anomalies by the decision tree models are then separated into each class of attack to be further classified. SVM-based SIDS is trained on the filtered datasets that should only contain anomaly data. The SVM algorithm mainly considers three hyperparameters: the C-value, the gamma-value, and the maximum number of iterations. The C-value is the maximum margin length around the hyperplane; the smaller C-value causes a larger margin space, which is usually accompanied by a decrease in accuracy. The gamma-value is the radius of influence each datapoint can have on clustering. The maximum number of iterations is set not only to prevent over-fitting of the model, but also to limit the total running time of the model. Using *GridSearchCV*, the optimized hyperparameter values are generated. The result is demonstrated in the following lines:

Result comparisons among different training algorithms and datasets				
Dataset	Training Algorithm	Experimentation Result	Referenced Result	Differences
KDD-CUP 1999	Decision Tree	99.91%	94.40% [3]	Data Preprossessing; Use of Feature Reduction Technique
KDD-CUP 1999	Decision Tree	99.91%	99.4% to 99.8% [5]	Use of Feature Reduction Technique
KDD-CUP 1999	Decision Tree	99.91%	81.05% [16]	Use of Feature Reduction Technique; Hyperparameter; Used for Classification
CICIDS-2017	Decision Tree	97.36%	99.60 to 99.80% [13]	Data Preprocessing; Use of Feature Reduction Technique; Hyperparameter
CICIDS-2017	Multiple Linear Regression	86.84%	97.86% and 73.79% [2]	Data Preprossessing; Different Feature Reduction Technique

Table 13: Table of comparisons between referenced literature and result obtained from experimentation

Class	Precision	Recall
DoS	100%	100%
Probing	99.04%	99.20%
R2L	98.59%	100.00%
U2R	100.00%	100.00%

Table 14: Table containing accuracy analysis on the classification of each attack class in *KDD-CUP 199*

Class	Precision	Recall
pod	100.00%	100.00%
smurf	100.00%	100.00%
teardrop	96.09%	100.00%
ipsweep	99.64%	99.96%
nmap	99.79%	98.13%

Table 15: Table containing accuracy analysis on the DoS/DDoS attack type on the DoS attack class in *KDD-CUP 199*

KDD-CUP 1999's model: C = 200, gamma = 0.01, max_iter = 100. *KDD-CUP 1999* DoS class's model: C = 200, gamma = 0.01, max_iter = 500. *CICIDS-2017*'s model: C = 100, gamma = 0.1, max_iter = 1500.

In the *KDD-CUP 1999* SVM training, the algorithm obtains an overall accuracy score of 99.98%, and an accuracy score of 100% on further classification of each DoS/DDoS attack type in the DoS class of the dataset. Table 14 shows the classification results of each attack class in *KDD-CUP 1999*, while Table 15 demonstrates the accuracy analysis of each DoS/DDoS attack type in the DoS class specifically. The results show a nicely performing algorithm that correctly classifies each class of attack and specific DoS attacks. However, the times for model construction and model fitting are also much higher than the algorithms proposed in Subsection 5.5, which are 29.95 and 3.10 seconds respectively. The further classification of each attack type in DoS class takes 66.32 seconds of construction time and 4.18 seconds of fitting time.

The result analysis for the SVM model on *CICIDS-2017* also exhibits great accuracy. An accuracy score of 99.98% is obtained, while Table 16 manifests excellent precision and recall on the classification of each DoS/DDoS attack class. The construction time is also larger than the AIDS algorithms as expected: 24.76 seconds; the fitting time is slightly over 1 second: 1.55 seconds.

5.6.2 Result and Performance Analysis

The SVM-based SIDS module is experimented with to output significantly greater accuracy scores than the decision tree-based AIDS module, or the multiple linear regression-based AIDS

Class	Precision	Recall
Bot	100.00%	99.75%
DoS slowloris	99.74%	99.32%
DoS Slowhttptest	99.72%	99.72%
DoS Hulk	99.99%	100.00%
DoS GoldenEye	99.70%	99.80%
DDoS	100.00%	100.00%

Table 16: Table containing accuracy analysis on the classification of each attack class in *CICIDS-2017*

Result comparisons among different datasets			
Dataset	Experimentation Result	Referenced Result	Differences
KDD-CUP 1999	99.98%	87.62% [3]	Data trained on; Hyperparameters
KDD-CUP 1999	99.05% (DoS); 0.91% (Probing); 0.04% (R2L); 0.00% (U2R)	99.64% (Benign); 98.55% (Probing); 98.92% (DoS); 63.20% (R2L); 59.60% (U2R) [10]	Data trained on; Hyperparameters
KDD-CUP 1999	99.98%	99.22% [7]	Data trained on; Algorithm Differences
CICIDS-2017	99.98%	99.32% [14]	Feature Reduction Technniques; Data trained on

Table 17: Table of comparisons between the referenced papers and this research project on SVM accuracy

module. The trade-off is a longer construction and fitting time. The decision tree algorithm uses a computational complexity of $O(n_{samples} \times log(n_{samples}) \times n_{PC})$, while the SVM algorithm outputs computational time between $\Omega(n_{samples}^2 \times n_{PC})$ and $O(n_{samples}^3 \times n_{PC})$. According to the framework, there will be an alarm after the decision tree algorithm detects a network anomaly, and then the process will be an automatic classification of which specific DoS/DDoS attack type for better mitigation on different network layers. Therefore, although the construction time of the SVM algorithm is dramatically larger, the fitting time is only slightly bigger than anomaly detection algorithms in terms of actual usage.

In comparisons with other researches done on the SVM algorithm, this research outputs a significantly higher result. The main reason is that all training and testing are performed on anomaly data only due to its characteristics as a SIDS. However, the listed referenced works uses SVM algorithm directly for classification or detection, with the loosely connected benign traffic associated with the abnormal traffic by a single radial basis function, outputting lower overall accuracy scores as shown in Table 17. For the comparison with [10] specifically, the data trained on causes substantial differences of the accuracy score on each class, for this research project did not attempt to evenly distribute the data according to each label. Hence, using accuracy score as measuring instrument, the attack class DoS, containing the most labeled data, naturally outputs the highest accuracy score.

6 Future Work

In the future, we believe it is important to conduct more research on other classification-based supervised machine learning algorithms, including K-Nearest Neighbors and Naïve Bayes, which are said to generate accurate results in some referenced works, to also test their performances on the datasets.

Testing the conclusion reached in this paper on other datasets, *UNSW-NB-15* is also an indispensable step to take to make the experimentation process more complete and further prove the effectiveness of our proposed HIDS framework.

7 Conclusion

In summary, we establish an HIDS framework to detect attacks with optimized efficiency using supervised machine learning algorithm based on two well-known datasets, *KDD-CUP 1999* and *CICIDS-2017*. The framework is composed of a feature reduction process, an AIDS component to detect network anomalies, and a SIDS component to classify the network anomalies. While selecting a proper algorithm to produce an accurate detector for AIDS, two algorithms, decision tree and multiple linear regression, are experimented with. As a result, decision tree produces higher accuracy score with low fitting time and tolerable construction time. On the other hand, the SVM algorithm is used as the SIDS classifier.

In the KDD-CUP 1999 dataset, the 41 features contained in the *KDD-CUP 1999* dataset are first expanded to 123 features through the spreading-out of categorical features, including "logged_in," "root_shell," "su_attempted," "is_host_login," "is_guest_login," and "land." Then, through the utilization of PCA, the 123 expanded feature set is extracted to 4 variables and 6 variables respectively for the decision tree algorithm and the multiple linear regression algorithm. Using a multiple linear regression model, an accuracy score of 98.96% is obtained for *KDD-CUP 1999*'s model. On the other hand, using the decision tree model, the *KDD-CUP 1999* dataset displays a roughly similar accuracy score of 99.91%.

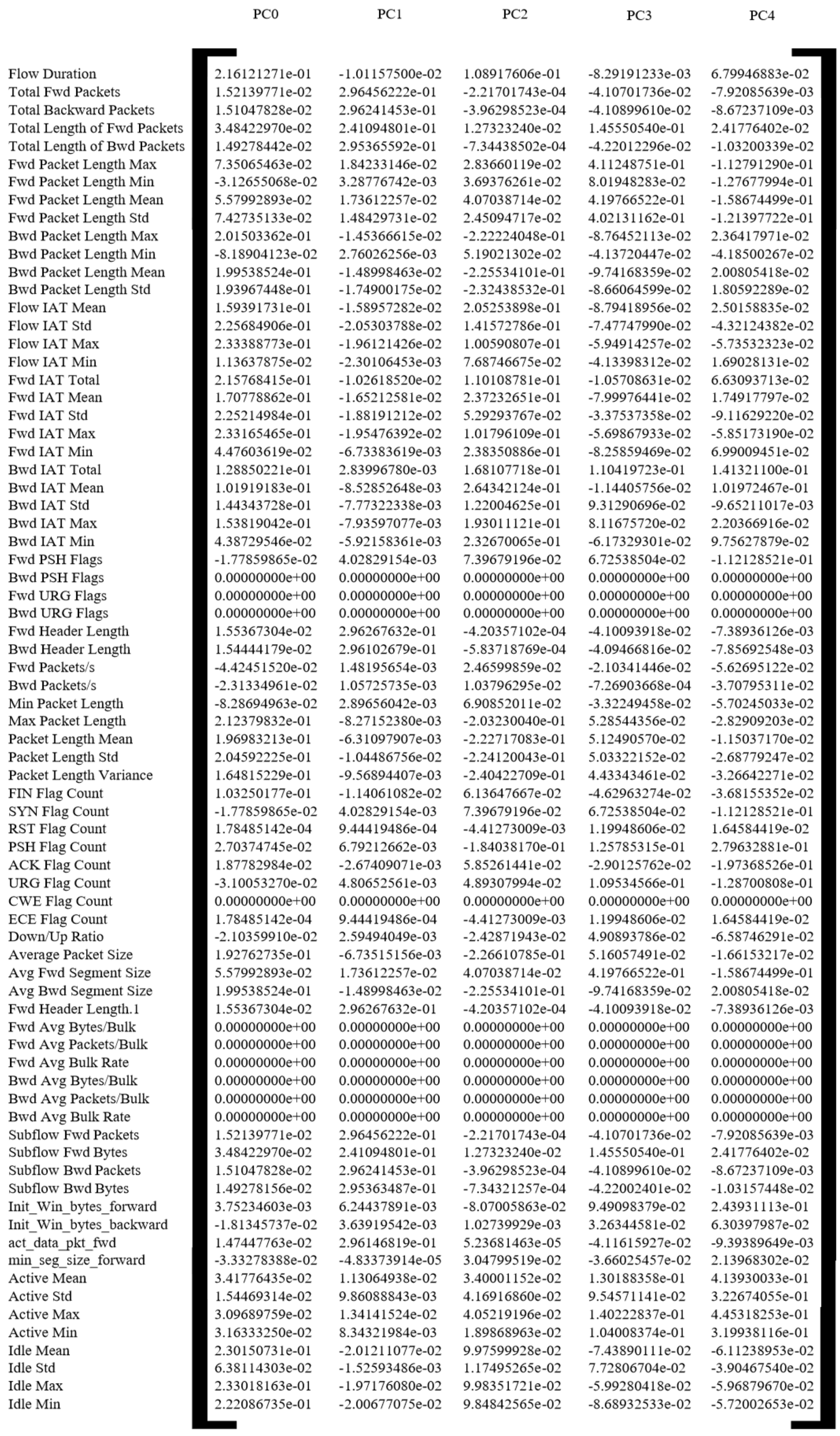
On the other hand, the two AIDS-candidate machine learning algorithms output different accuracy scores in the CICIDS-2017 dataset. The preprocessing of data in CICIDS-2017 include expanding the categorical features ("Destination Port") are expanded to become a set of numerical features. PCA techniques are used to reduce the number of features down to 5 and 7 principal components for decision tree and multiple linear regression respectively. As a result, using multiple linear regression, an accuracy score of 86.84% is outputted using 7 principal components, while the decision tree model returns a higher accuracy score of 97.36% with only 5 components. Therefore, due to the inconsistency in accuracy scores of the multiple linear regression models for the two datasets, we conclude that decision trees are more accurate models for anomaly detection, with less risks of wrongly predicting the labeled data, with acceptable construction time and low fitting time.

The SVM-based SIDS performs nicely on its job of classifying specific network attack types based on the data predicted as anomaly decision tree. The SIDS module takes significantly more construction time, which can cause a long training time of the SVM algorithm on other datasets, as mentioned in Section 6. A higher construction time of the model usually accompanies with a more sophisticated high-dimensional splitting curve, also leading to higher fitting time and causing problems in real use of this hybrid framework. After optimizing the hyperparameters in the SVM algorithm with respect to both the construction time and fitting time, the total running times for both datasets are reduced significantly. The accuracy scores are 99.98% and 99.98% for *KDD-CUP 1999* and *CICIDS-2017* respectively. Still, it uses remarkably more time than the AIDS component. Using this framework, the usual traffic will not alarm the SIDS component, so there will be no unnecessary time spent on the classification of anomalies when there is none. Having compressed the fitting time of SVM into at most 5 seconds for both datasets, in the scenario when there is a live DoS/DDoS attack, the framework will also quickly generate a mitigation method using the constructed frameworks containing known types of DoS/DDoS attacks from the two well-known datasets. The SIDS component assists specific mitigation of the framework by providing information about the network layers the attacker is mainly targeting.

The hybrid framework can be summarized into three key parts: feature reduction, AIDS, and SIDS. The models generated in each part are stored as following. Figure 23 and Figure 24 are the matrices of feature importance, the transfer matrix P^T in Section 3. Figure 25, and Figure 26 are the explained variance ratios by PCA for both datasets. The PCA matrices are stored as constant values to directly apply feature transformations on new data. The topologies of the decision trees models for each dataset are stored as PKL files for convenient loading and fitting in real scenarios. The SVM models built for classification on the known attacks are also stored as two separate PKL files. The proposed HIDS framework potentially guarantees high accuracy and low running time in real-time network situations.

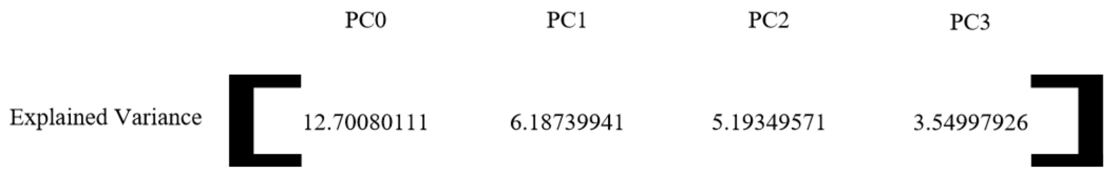
	PC0	PC0	PC1	PC2
duration	1.19891822e-02	1.93495388e-02	-2.63887248e-02	1.41450959e-02
src_bytes	3.39518346e-04	1.00607694e-03	-9.60772015e-04	4.27854198e-04
dst_bytes	3.85005934e-04	1.50772372e-03	-9.30170479e-04	1.68272952e-04
wrong_fragment	1.79406250e-03	2.21853287e-03	-3.68689429e-04	-4.54854309e-05
urgent	2.84689809e-04	2.12507006e-03	2.24429516e-03	2.57764901e-03
hot	4.63345539e-03	3.19486812e-02	3.68292683e-02	4.61878527e-01
num_failed_logins	1.07908111e-03	4.18661219e-03	1.08899599e-04	9.16514948e-03
num_compromised	5.87024854e-04	5.20461921e-03	5.81221481e-03	6.05445453e-03
num_root	7.47523457e-04	6.08353766e-03	6.62716054e-03	6.08223316e-03
num_file_creations	1.71035104e-03	8.45185392e-03	6.83379098e-03	4.07306164e-03
num_shells	1.02941320e-03	6.43854066e-03	6.01142895e-03	-4.21502114e-04
num_access_files	2.36213180e-03	2.49421651e-02	2.34967210e-02	-1.99269455e-03
num_outbound_cmds	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
count	-1.92182541e-01	-2.28572071e-01	-1.48080250e-01	2.44717403e-02
srv_count	-2.44131526e-01	-1.58996029e-01	-9.93135521e-02	1.87953899e-02
serror_rate	2.28529399e-01	-2.06343730e-01	9.67482550e-02	-4.70569041e-03
srv_serror_rate	2.28370065e-01	-2.06532950e-01	9.74766010e-02	-4.75708864e-03
rerror_rate	9.83110957e-02	1.84390247e-01	-3.50577918e-01	8.72784452e-03
srv_rerror_rate	9.80006039e-02	1.84965100e-01	-3.50360107e-01	8.73803128e-03
same_srv_rate	-2.56475302e-01	1.19184932e-01	7.53935430e-02	-4.70231272e-03
diff_srv_rate	1.13841713e-01	-5.03127106e-03	-9.31442779e-02	1.86135291e-02
srv_diff_host_rate	1.47866402e-02	1.63651425e-01	1.00920825e-01	-3.88935432e-02
dst_host_count	-2.70284624e-02	-2.77295292e-01	-1.46283243e-01	3.06462164e-02
dst_host_srv_count	-2.59650023e-01	6.16811795e-02	5.61453477e-02	-3.00731455e-02
dst_host_same_srv_rate	-2.59616604e-01	8.34482386e-02	6.34613827e-02	-2.89816362e-02
dst_host_diff_srv_rate	1.01458987e-01	2.55947964e-02	-9.52108282e-02	2.67036414e-02
dst_host_same_src_port_rate	-2.43432868e-01	-1.27417596e-01	-1.06620789e-01	1.84316913e-02
dst_host_srv_diff_host_rate	1.51317564e-02	1.42838132e-01	1.47777614e-02	-2.96377559e-02
dst_host_serror_rate	2.28580570e-01	-2.06155699e-01	9.68734488e-02	-4.64653329e-03
dst_host_srv_serror_rate	2.28355535e-01	-2.06787433e-01	9.72278953e-02	-4.68719574e-03
dst_host_rerror_rate	9.83010184e-02	1.85782572e-01	-3.47771385e-01	8.74864620e-03
dst_host_srv_rerror_rate	9.81558035e-02	1.85627357e-01	-3.48303972e-01	8.76918843e-03
logged_in_0	-2.24731595e-02	-2.95697366e-01	-2.59385213e-01	2.51274324e-02
logged_in_1	2.24731595e-02	2.95697366e-01	2.59385213e-01	-2.51274324e-02
root_shell_0	-1.14036673e-03	-1.17236990e-02	-1.20410164e-02	-9.46825661e-03
root_shell_1	1.14036673e-03	1.17236990e-02	1.20410164e-02	9.46825661e-03
su_attempted_0	-1.26574745e-03	-9.20269802e-03	-9.80250050e-03	-8.46072203e-03
su_attempted_1	5.64091547e-04	3.81168601e-03	3.93676658e-03	2.95635348e-03
su_attempted_2	1.15451456e-03	8.60405466e-03	9.25432943e-03	8.30791578e-03
is_host_login_0	-2.98860780e-04	-3.92358626e-04	-5.18882234e-04	-5.32324747e-04
is_host_login_1	2.98860780e-04	3.92358626e-04	5.18882234e-04	5.32324747e-04
land_0	-9.50994327e-04	-1.44576757e-03	-1.97421069e-03	1.40593096e-03
land_1	9.50994327e-04	1.44576757e-03	1.97421069e-03	-1.40593096e-03
is_guest_login_0	-5.32997797e-03	-3.45379466e-02	-4.02953096e-02	-5.16123037e-01
is_guest_login_1	5.32997797e-03	3.45379466e-02	4.02953096e-02	5.16123037e-01
protocol_type_icmp	-2.44799910e-01	-1.53980048e-01	-9.85863577e-02	1.80060086e-02
protocol_type_tcp	2.43550115e-01	1.40896050e-01	9.68139028e-02	-1.80491242e-02
protocol_type_udp	1.30853007e-02	3.88820051e-02	8.43294177e-03	-6.25893988e-04
service_IRC	3.11247675e-03	1.01834532e-02	-6.34542993e-03	7.12593507e-04
service_X11	9.58398445e-04	2.96496575e-03	1.14608326e-03	-2.88656835e-04
service_Z39_50	6.95271753e-03	-3.43799218e-03	-2.02356070e-03	5.60823510e-05
service_aol	2.94519482e-04	3.48751062e-04	-1.10542015e-03	1.18328068e-04
service_auth	7.21192251e-03	9.67073234e-03	6.58060522e-03	-1.88433989e-03
service_bgp	6.81704200e-03	-3.31244975e-03	-2.07419830e-03	6.29602108e-05
service_courier	6.82398137e-03	-3.42519247e-03	-9.86076403e-04	-5.22736399e-05
service_csnet_ns	6.82940800e-03	-3.57919035e-03	-1.65982411e-03	2.91121856e-05
service_ctf	6.94125817e-03	-3.22572657e-03	-2.03115444e-03	1.29924906e-05
service_daytime	6.81509831e-03	-3.31614871e-03	-2.04339727e-03	3.98217250e-04
service_discard	6.89689405e-03	-3.43100352e-03	-1.93437410e-03	3.66637099e-04
service_domain	6.83779521e-03	-3.09762171e-03	-1.53082956e-03	3.26923246e-06
service_domain_u	6.72592420e-04	3.17306639e-02	2.37142669e-02	-8.83927241e-03
service_echo	6.93534333e-03	-3.57161790e-03	-1.71248879e-03	3.42309652e-04
service_eco_i	-2.56830514e-03	3.77516010e-02	1.76251579e-02	-1.38109818e-02
service_ecr_i	-2.44360428e-01	-1.58284104e-01	-1.00404004e-01	1.95074075e-02
service_efs	6.83837508e-03	-2.93623357e-03	-1.80577499e-03	-2.18964455e-05
service_exec	6.97190283e-03	-3.03051560e-03	-1.69776403e-03	3.11205122e-04
service_finger	8.83914756e-03	7.94329334e-03	6.45444878e-03	-2.26435142e-03
service_ftp	7.74880992e-03	3.06656381e-02	3.64423028e-02	4.87374481e-01
service_ftp_data	1.14864343e-02	4.80347563e-02	4.26232765e-02	-8.54969246e-03
service_gopher	6.96240515e-03	-3.06154882e-03	-2.38534425e-03	1.13317448e-05
service_harvest	2.92452765e-04	3.53188565e-04	-1.10833628e-03	1.34945647e-04
service_hostnames	6.82453287e-03	-3.53163665e-03	-1.83203292e-03	9.60711666e-05
service_http	2.32399742e-02	2.89124716e-01	1.84761270e-01	-6.33408585e-02
service_http_2784	2.93145982e-04	3.62219045e-04	-1.12634742e-03	1.05191741e-04
service_http_443	6.86828151e-03	-3.18911964e-03	-1.47593534e-03	-5.64684183e-05
service_http_8001	3.17042830e-04	-3.77528248e-05	-4.75513633e-04	1.04041438e-04
service_imap4	6.84944999e-03	-3.45193674e-03	-1.71127222e-03	2.50751595e-05
service_iso_tsap	6.86031898e-03	-3.23155873e-03	-2.29575502e-03	5.00284720e-05
service_klogin	6.96992802e-03	-3.07161302e-03	-1.67073911e-03	-1.36500408e-05
service_kshell	6.87991886e-03	-2.95335869e-03	-1.78507823e-03	-4.04692545e-05
service_ldap	6.79805860e-03	-3.15576557e-03	-1.58923123e-03	2.14394431e-05
service_link	6.89594295e-03	-3.08837463e-03	-2.30663965e-03	6.38236187e-05
service_login	6.96259426e-03	-3.23850822e-03	-1.31643946e-03	2.79572493e-04
service_mtp	6.94397381e-03	-3.07615871e-03	-2.34433354e-03	2.20201440e-05
service_name	6.94708548e-03	-3.15799642e-03	-2.23065207e-03	3.27952250e-05
service_netbios_dgm	6.86621207e-03	-3.55218682e-03	-1.79745127e-03	2.76078632e-05
service_netbios_ns	6.88119049e-03	-3.50465339e-03	-1.91224516e-03	5.66196432e-05
service_netbios_ssn	6.78259820e-03	-3.40947882e-03	-1.91826433e-03	2.34706999e-05
service_netstat	6.97549750e-03	-3.48752245e-03	-1.98794240e-03	7.36847239e-05
service_nnsp	6.96367411e-03	-3.21006208e-03	-1.56483375e-03	-2.60778811e-05
service_nntp	6.90013914e-03	-3.30760084e-03	-2.06141173e-03	4.18094654e-04
service_ntp_u	1.87916855e-03	6.47654669e-03	4.23167093e-03	-7.33039881e-04
service_other	2.82163747e-02	3.05734095e-02	-6.65298105e-02	1.75948917e-02
service_pm_dump	3.89516052e-04	3.12575380e-04	-5.30576451e-04	1.21303532e-04
service_pop_2	6.89521473e-03	-3.41792656e-03	-1.99658794e-03	5.19447579e-05
service_pop_3	6.65722083e-03	4.51844510e-03	3.75892366e-03	-9.86730540e-04
service_printer	6.87022105e-03	-3.02416234e-03	-1.62070465e-03	3.04822535e-04
service_private	2.41593156e-01	-1.11679274e-01	-6.03633074e-02	-1.45877160e-03
service_red_i	6.37030179e-05	2.12511428e-04	1.00476229e-04	-1.99345221e-05
service_remote_job	6.95822648e-03	-3.05261310e-03	-2.41845627e-03	4.28090015e-05
service_rje	6.88205285e-03	-2.88348308e-03	-2.63076030e-03	5.69483663e-05
service_shell	6.86802847e-03	-2.99753136e-03	-1.58536198e-03	3.24298316e-04
service_smtp	1.46708280e-02	1.05719733e-01	9.40092444e-02	-2.42297998e-02
service_sql_net	6.87712408e-03	-3.41842599e-03	-2.04996747e-03	5.29714372e-05
service_ssh	6.92828685e-03	-2.99903730e-03	-2.12986764e-03	7.68768193e-05
service_sunrpc	6.84252171e-03	-3.67904370e-03	-1.56021514e-03	3.80786069e-04
service_supdup	6.97594891e-03	-3.60624790e-03	-1.75646303e-03	2.84864092e-05
service_systat	6.92687325e-03	-3.46037075e-03	-2.05992543e-03	1.08768482e-04
service_telnet	9.13387425e-03	8.87857678e-03	1.0111368e-02	2.72693658e-03
service_tftp_u	1.44965618e-04	2.12409952e-04	1.09831936e-04	9.11090459e-06
service_tim_i	3.53650128e-08	3.87225002e-04	2.58164372e-04	-7.20077608e-05
service_time	6.06073741e-03	-1.94593738e-04	2.99099773e-04	-2.65346979e-04
service_urh_i	1.47429093e-04	8.86444171e-04	5.14509801e-04	-7.79733096e-05
service_urp_i	2.97965882e-03	1.55089887e-03	-1.57999371e-03	1.26373870e-03
service_uucp	6.91990507e-03	-3.02798297e-03	-1.66157783e-03	3.59077251e-04
service_uucp_path	6.89620322e-03	-3.34954859e-03	-2.10359481e-03	4.23045742e-05
service_vmnet	6.92167371e-03	-3.67571582e-03	-1.63487658e-03	5.41188202e-05
service_whois	6.90987074e-03	-3.09614063e-03	-2.30770502e-03	5.08127376e-05
flag_OTH	7.99753754e-04	1.02473490e-03	-5.53259460e-04	2.31586155e-03
flag_REJ	9.54578523e-02	1.80159702e-01	-3.41405399e-01	6.07972979e-03
flag_RSTO	1.28755946e-02	2.28927748e-02	-4.26242967e-02	1.08080332e-02
flag_RSTOS0	2.28403791e-03	6.95788161e-04	-3.69538396e-03	7.52222508e-04
flag_RSTR	1.59748723e-02	3.03963141e-02	-6.39904403e-02	7.28467370e-03
flag_S0	2.28106347e-01	-2.06942619e-01	9.70230395e-02	-4.78253021e-03
flag_S1	2.27998044e-03	6.98158368e-03	7.65944645e-03	1.57765851e-03
flag_S2	1.09885113e-03	4.08751295e-03	4.50331866e-03	1.15525893e-04
flag_S3	7.82215837e-04	2.07846216e-03	2.17850467e-03	-2.24947081e-04
flag_SF	-2.59515665e-01	8.49102530e-02	1.04998211e-01	-5.97280977e-04
flag_SH	6.76768643e-03	-4.52505089e-03	8.54442103e-04	1.03647162e-03

Figure 23: PCA transfer matrix for *KDD-CUP 1999* with 4 principal components.



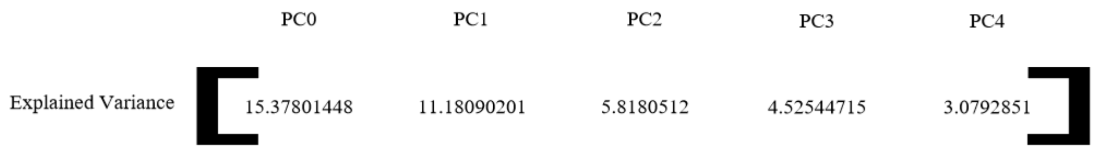
	PC0	PC1	PC2	PC3	PC4
Flow Duration	2.16121271e-01	-1.01157500e-02	1.08917606e-01	-8.29191233e-03	6.79946883e-02
Total Fwd Packets	1.52139771e-02	2.96456222e-01	-2.21701743e-04	-4.10701736e-02	-7.92085639e-03
Total Backward Packets	1.51047828e-02	2.96241453e-01	-3.96298523e-04	-4.10899610e-02	-8.67237109e-03
Total Length of Fwd Packets	3.48422970e-02	2.41094801e-01	1.27323240e-02	1.45550540e-01	2.41776402e-02
Total Length of Bwd Packets	1.49278442e-02	2.95365592e-01	-7.34438502e-04	-4.22012296e-02	-1.03200339e-02
Fwd Packet Length Max	7.35065463e-02	1.84233146e-02	2.83660119e-02	4.11248751e-01	-1.12791290e-01
Fwd Packet Length Min	-3.12655068e-02	3.28776742e-03	3.69376261e-02	8.01948283e-02	-1.27677994e-01
Fwd Packet Length Mean	5.57992893e-02	1.73612257e-02	4.07038714e-02	4.19766522e-01	-1.58674499e-01
Fwd Packet Length Std	7.42735133e-02	1.48429731e-02	2.45094717e-02	4.02131162e-01	-1.21397722e-01
Bwd Packet Length Max	2.01503362e-01	-1.45366615e-02	-2.22224048e-01	-8.76452113e-02	2.36417971e-02
Bwd Packet Length Min	-8.18904123e-02	2.76026256e-03	5.19021302e-02	-4.13720447e-02	-4.18500267e-02
Bwd Packet Length Mean	1.99538524e-01	-1.48998463e-02	-2.25534101e-01	-9.74168359e-02	2.00805418e-02
Bwd Packet Length Std	1.93967448e-01	-1.74900175e-02	-2.32438532e-01	-8.66064599e-02	1.80592289e-02
Flow IAT Mean	1.59391731e-01	-1.58957282e-02	2.05253898e-01	-8.79418956e-02	2.50158835e-02
Flow IAT Std	2.25684906e-01	-2.05303788e-02	1.41572786e-01	-7.47747990e-02	-4.32124382e-02
Flow IAT Max	2.33388773e-01	-1.96121426e-02	1.00590807e-01	-5.94914257e-02	-5.73532323e-02
Flow IAT Min	1.13637875e-02	-2.30106453e-03	7.68746675e-02	-4.13398312e-02	1.69028131e-02
Fwd IAT Total	2.15768415e-01	-1.02618520e-02	1.10108781e-01	-1.05708631e-02	6.63093713e-02
Fwd IAT Mean	1.70778862e-01	-1.65212581e-02	2.37232651e-01	-7.99976441e-02	1.74917797e-02
Fwd IAT Std	2.25214984e-01	-1.88191212e-02	5.29293767e-02	-3.37537358e-02	-9.11629220e-02
Fwd IAT Max	2.33165465e-01	-1.95476392e-02	1.01796109e-01	-5.69867933e-02	-5.85173190e-02
Fwd IAT Min	4.47603619e-02	-6.73383619e-03	2.38350886e-01	-8.25859469e-02	6.99009451e-02
Bwd IAT Total	1.28850221e-01	2.83996780e-03	1.68107718e-01	1.10419723e-01	1.41321100e-01
Bwd IAT Mean	1.01919183e-01	-8.52852648e-03	2.64342124e-01	-1.14405756e-02	1.01972467e-01
Bwd IAT Std	1.44343728e-01	-7.77322338e-03	1.22004625e-01	9.31290696e-02	-9.65211017e-03
Bwd IAT Max	1.53819042e-01	-7.93597077e-03	1.93011121e-01	8.11675720e-02	2.20366916e-02
Bwd IAT Min	4.38729546e-02	-5.92158361e-03	2.32670065e-01	-6.17329301e-02	9.75627879e-02
Fwd PSH Flags	-1.77859865e-02	4.02829154e-03	7.39679196e-02	6.72538504e-02	-1.12128521e-01
Bwd PSH Flags	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Fwd URG Flags	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Bwd URG Flags	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Fwd Header Length	1.55367304e-02	2.96267632e-01	-4.20357102e-04	-4.10093918e-02	-7.38936126e-03
Bwd Header Length	1.54444179e-02	2.96102679e-01	-5.83718769e-04	-4.09466816e-02	-7.85692548e-03
Fwd Packets/s	-4.42451520e-02	1.48195654e-03	2.46599859e-02	-2.10341446e-02	-5.62695122e-02
Bwd Packets/s	-2.31334961e-02	1.05725735e-03	1.03796295e-02	-7.26903668e-04	-3.70795311e-02
Min Packet Length	-8.28694963e-02	2.89656042e-03	6.90852011e-02	-3.32249458e-02	-5.70245033e-02
Max Packet Length	2.12379832e-01	-8.27152380e-03	-2.03230040e-01	5.28544356e-02	-2.82909203e-02
Packet Length Mean	1.96983213e-01	-6.31097907e-03	-2.22717083e-01	5.12490570e-02	-1.15037170e-02
Packet Length Std	2.04592225e-01	-1.04486756e-02	-2.24120043e-01	5.03322152e-02	-2.68779247e-02
Packet Length Variance	1.64815229e-01	-9.56891407e-03	-2.40422709e-01	4.43343461e-02	-3.26642271e-02
FIN Flag Count	1.03250177e-01	-1.14061082e-02	6.13647667e-02	-4.62963274e-02	-3.68155352e-02
SYN Flag Count	-1.77859865e-02	4.02829154e-03	7.39679196e-02	6.72538504e-02	-1.12128521e-01
RST Flag Count	1.78485142e-04	9.44419486e-04	-4.41273009e-03	1.19948606e-02	1.64584419e-02
PSH Flag Count	2.70374745e-02	6.79212662e-03	-1.84038170e-01	1.25785315e-01	2.79632881e-01
ACK Flag Count	1.87782984e-02	-2.67409071e-03	5.85261441e-02	-2.90125762e-02	-1.97368526e-01
URG Flag Count	-3.10053270e-02	4.80652561e-03	4.89307994e-02	1.09534566e-01	-1.28700808e-01
CWE Flag Count	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
ECE Flag Count	1.78485142e-04	9.44419486e-04	-4.41273009e-03	1.19948606e-02	1.64584419e-02
Down/Up Ratio	-2.10359910e-02	2.59494049e-03	-2.42871943e-02	4.90893786e-02	-6.58746291e-02
Average Packet Size	1.92762735e-01	-6.73515156e-03	-2.26610785e-01	5.16057491e-02	-1.66153217e-02
Avg Fwd Segment Size	5.57992893e-02	1.73612257e-02	4.07038714e-02	4.19766522e-01	-1.58674499e-01
Avg Bwd Segment Size	1.99538524e-01	-1.48998463e-02	-2.25534101e-01	-9.74168359e-02	2.00805418e-02
Fwd Header Length.1	1.55367304e-02	2.96267632e-01	-4.20357102e-04	-4.10093918e-02	-7.38936126e-03
Fwd Avg Bytes/Bulk	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Fwd Avg Packets/Bulk	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Fwd Avg Bulk Rate	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Bwd Avg Bytes/Bulk	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Bwd Avg Packets/Bulk	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Bwd Avg Bulk Rate	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00	0.00000000e+00
Subflow Fwd Packets	1.52139771e-02	2.96456222e-01	-2.21701743e-04	-4.10701736e-02	-7.92085639e-03
Subflow Fwd Bytes	3.48422970e-02	2.41094801e-01	1.27323240e-02	1.45550540e-01	2.41776402e-02
Subflow Bwd Packets	1.51047828e-02	2.96241453e-01	-3.96298523e-04	-4.10899610e-02	-8.67237109e-03
Subflow Bwd Bytes	1.49278156e-02	2.95363487e-01	-7.34321257e-04	-4.22002401e-02	-1.03157448e-02
Init_Win_bytes_forward	3.75234603e-03	6.24437891e-03	-8.07005863e-02	9.49098379e-02	2.43931113e-01
Init_Win_bytes_backward	-1.81345737e-02	3.63919542e-03	1.02739929e-03	3.26344581e-02	6.30397987e-02
act_data_pkt_fwd	1.47447763e-02	2.96146819e-01	5.23681463e-05	-4.11615927e-02	-9.39389649e-03
min_seg_size_forward	-3.33278388e-02	-4.83373914e-05	3.04799519e-02	-3.66025457e-02	2.13968302e-02
Active Mean	3.41776435e-02	1.13064938e-02	3.40001152e-02	1.30188358e-01	4.13930033e-01
Active Std	1.54469314e-02	9.86088843e-03	4.16916860e-02	9.54571141e-02	3.22674055e-01
Active Max	3.09689759e-02	1.34141524e-02	4.05219196e-02	1.40222837e-01	4.45318253e-01
Active Min	3.16333250e-02	8.34321984e-03	1.89868963e-02	1.04008374e-01	3.19938116e-01
Idle Mean	2.30150731e-01	-2.01211077e-02	9.97599928e-02	-7.43890111e-02	-6.11238953e-02
Idle Std	6.38114303e-02	-1.52593486e-03	1.17495265e-02	7.72806704e-02	-3.90467540e-02
Idle Max	2.33018163e-01	-1.97176080e-02	9.98351721e-02	-5.99280418e-02	-5.96879670e-02
Idle Min	2.22086735e-01	-2.00677075e-02	9.84842565e-02	-8.68932533e-02	-5.72002653e-02

Figure 24: PCA transfer matrix for *CICIDS-2017* with 5 principal components.



	PC0	PC1	PC2	PC3
Explained Variance	12.70080111	6.18739941	5.19349571	3.54997926

Figure 25: PCA explained variance ratios for *KDD-CUP 1999* with 4 principal components.



	PC0	PC1	PC2	PC3	PC4
Explained Variance	15.37801448	11.18090201	5.8180512	4.52544715	3.0792851

Figure 26: PCA explained variance ratios for *CICIDS-2017* with 5 principal components.

References

[1] "Ddos attack trends for 2020," F5 Application Threat Intelligence, May 2021. [Online]. Available: https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020

[2] S. Swathi and G. Lakshmeeswari, "A machine learning approach for ddos (distributed denial of service) attack detection using multiple linear regression," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 63, no. 1, 2020, p. 51.

[3] A. Ahmed *et al.*, "An intelligent and time-efficient ddos identification framework for real-time enterprise networks: Sad-f: Spark based anomaly detection framework," vol. 8, no. 1, p. 219483, 2020.

[4] "The caida "ddos attack 2007" dataset," Feb 2010. [Online]. Available: https://www.caida.org/catalog/datasets/ddos-20070804_dataset

[5] I. Cvitić *et al.*, "An overview of distributed denial of service traffic detection approaches," in *Promet - Traffic Transportation*, vol. 31, no. 8, 2019, p. 453.

[6] J.-H. Lee *et al.*, "Effective value of decision tree with kdd 99 intrusion detection datasets for intrusion detection system."

[7] N. A. Awad, "Enhancing network intrusion detection model using machine learning algorithms," in *Department of Computer and Information Systems*, vol. 67, no. 1, 2021, pp. 979–990.

[8] "Kdd cup 1999 data," Oct 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[9] Y. Bouzida *et al.*, "Efficient intrusion detection using principal component analysis."

[10] S. Singh *et al.*, "Improved support vector machine for cyber attack detection."

[11] "Intrusion detection evaluation dataset (cic-ids2017)," 2017. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[12] Y. Li and L. Guo, "An active learning based tcm-knn algorithm for supervised network intrusion detection," in *Computer and Security*, vol. 26, no. 7, 2007, p. 459.

[13] Kurniabudi *et al.*, "Cicids-2017 dataset feature analysis with information gain for anomaly detection," vol. 8, pp. 132 911–132 921, 2020.

[14] S. Allagi *et al.*, "A robust support vector machine based auto-encoder for dos attacks identification in computer networks," 2021.

[15] F. S. Alraddadi *et al.*, "Impact of minority class variability on anomaly detection by means of random forests and support vector machines," pp. 416–428, August 2021.

[16] R. R. Devi and M. Abualkibash, "Intrusion detection system classification using different machine learning algorithms on kdd-99 and nsl-kdd datasets - a review paper," in *International Journal of Computer Science Information Technology (IJCSIT)*, vol. 11, no. 3, 2019, p. 65.

[17] W. Wang and R. Battiti, "Identifying intrusions in computer networks with principal component analysis," 2006.

[18] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review," vol. 30, no. 6, p. 353, 2011.

[19] A. Khraisat *et al.*, "Hybrid intrusion detection system based on the stacking ensemble of c5 decision tree classifier and one class support vector machine," 2020.

[20] H. Venkatnarayanan and V. Bhanumathi, "Automatic cataract classification system," 04 2016, pp. 0815–0819.

[21] "Thinkpad x1 carbon gen 8," 2021. [Online]. Available: https://www.lenovo.com/hk/en/laptops/thinkpad/thinkpad-x1/X1-Carbon-Gen-8-/p/22TP2X1X1C8

[22] Yang *et al.*, "Improving the detection rate of rarely appearing intrusions in network-based intrusion detection systems," in *Computers, Materials Continua*, vol. 66, no. 2, 2021, p. 1647.

[23] "Decision trees," Scikit Learn. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html

[24] "Linear regression," Statsmodels. [Online]. Available: https://www.statsmodels.org/stable/regression.html